Universidad Carlos III de Madrid
Escuela Politécnica Superior

Proyecto de fin de carrera

Entorno de monitorización de sistemas informáticos

embarcados mediante pantallas táctiles.

Versión: 1.0

Creación: Diciembre 2010

Autor: Vicente García Adánez

Tutor: Juan Llorens Morillo

Esta página ha sido dejada en blanco intencionadamente.

RESUMEN:

Este proyecto ha sido realizado para ser presentado como proyecto de fin de carrera en colaboración con la empresa finlandesa Mobile Net Control (MNC de ahora en adelante) con sede en Mariehamn, capital de las islas Åland, provincia de Finlandia, gracias a la obtención de una beca Erasmus para el año 2010.

MNC en una de sus áreas provee a sus clientes de ordenadores para sus barcos creando un sistema de comunicación en ellos. Además de mantenerlos funcionando correctamente se encarga de que la información y configuración en los equipos no se pierda y para ello realiza a través de internet copias de seguridad de sus archivos.

Como utilidad para sus barcos ha creído conveniente desarrollar una aplicación software fácil e intuitiva desde la que se pueda monitorizar el estado de los ordenadores que pueda tener su barco. Ésta se encargará de comprobar que funcionan correctamente y en caso de que no lo hagan ayudará a solucionarlo.

El objetivo de este proyecto es por tanto la realización de una aplicación para pantalla táctil que permita monitorizar el estado de diferentes componentes en un barco desde el lugar donde esté instalada. Su uso está destinado a la persona responsable de este cometido, que bien puede ser un capitán como cualquier otra persona del barco no relacionada con la informática.

PROYECTO DE FIN DE CARRERA (PFC).

Juan Llorens Morillo, como profesor del Departamento de Informática de la universidad Carlos III de Madrid, adscrito a la escuela politécnica superior,

CERTIFICA: que el presente Proyecto Fin de Carrera, titulado "Entorno de monitorización de sistemas informáticos embarcados mediante pantallas táctiles", ha sido desarrollado en colaboración con la empresa Mobile Net Control y bajo su dirección en la Escuela Politécnica de la Universidad Carlos III de Madrid por Vicente García Adánez para obtener el título de Ingeniero Técnico en Informática de Gestión.

Considerando este Proyecto Fin de Carrera finalizado, autorizan su presentación y defensa.

Y para que aquí conste, firma en Leganés a 23 de noviembre de 2010.

Fdo.: Juan Llorens Morillo

AGRADECIMIENTOS.

Puede que esta sea una de las partes más difíciles del proyecto, hay tanta gente que me ha ayudado a conseguir esto que siento mucho si no sale vuestro nombre aquí, ya me daré de golpes contra la pared cuando me acuerde.

Éste es un éxito de mi vida que quiero brindar a mis personas más allegadas y en primer y más importante lugar a mi madre, de cuya tripa vengo. Sin ella nada de esto habría sido posible, ya no sólo porque me aconsejara vivir cuando yo era tan pequeño que ni siquiera puedo acordarme, sino porque me ha enseñado a enfrentarme a ella como nadie. Tras haber pasado (qué gusto decirlo así) su enfermedad, ha sido mi mejor ejemplo de superación y quien me ha dado este año la mejor lección que nadie me podía dar nunca, ojalá algún día pueda ser como ella en infinidad de aspectos.

Gracias a mi abuela Adela, porque este proyecto en sí está basado en la plataforma .NET, pero ella es quien ha mantenido la plataforma de mi vida en orden, ha sido el pilar de cariño que todos necesitamos y que no todos tienen. Mirando un poco más arriba quiero dar infinitas gracias a mi querido abuelo Manolo de quien tanto aprendí, porque aunque él ya no esté aquí sería a la primera persona que iría a abrazar y decir: “¡He acabado el proyecto!”; y que nadie dude de que se lo diré aunque me agote de subir escaleras.

Gracias a mi tío Manu por ser el trocito de serenidad que nunca he tenido y mi “pepito grillo” particular, sin su apoyo puedo asegurar que no estaría aquí.

Gracias a mi familia, a mi hermana Clara por ser un gran ejemplo de superación, a mi hermana Liv, mi tía Adela, mis primos Edu y Ade a quienes quiero como hermanos, a mi tía Cele y a mi abuela Lela (no me olvido de Lelo, a quien también quiero brindar este éxito).

Gracias también a mi padre por enseñarme qué no hacer y por moldear a golpe de martillo una personalidad férrea.

Gracias a mis amigos langostinos, los más cercanos y familiares por hacer de mis ratos libres los mejores de la historia: Adri, Alvarito, Chema, Deivid, Gian, Mena y Raúl. Ellos son una de las partes más importantes de mi vida.

Gracias al resto de amigos del mundo entero por haberme apoyado siempre y en especial a una persona, Liz, por ser el rayito de luz que no ha dejado de brillar en los días más oscuros. Tampoco puedo olvidarme de Richi, todavía le debo muchas horas y el íntegro reembolso del precio de unas cuantas asignaturas. Pero de todos estos años en Colmenarejo quiero resaltar la figura de una persona muy especial, Carla, mi “ángel de la guarda” durante esta carrera. Porque todos nos sentimos sin fuerzas en algún momento y yo, de verdad, todavía no sé que vio en mí para creer de esa manera. Sin apenas conocerme me arropó con una manta a la que abrió un agujerito por donde pude respirar, me hizo un lazo bonito y me echó a su hombro; así me llevó por nuestro camino. Todo con esa sonrisa maravillosa, es algo que nunca podré entender.

Gracias de todo corazón a Paloma Soler, quien me vio crecer desde el principio y quien me ha seguido el rastro desde que empecé mi vida adulta y esta carrera y muchas otras hasta el día de hoy, porque realmente sin sus consejos nunca habría podido llevar a cabo ni cerrar este capítulo. Presente cuando empezó hace unos años pero no cuando acabó, sólo tengo las más sinceras palabras de agradecimiento hacia ella. Sin su ayuda no conocería ni qué ha sido la felicidad ni qué es el dolor más profundo.

Gracias también a todos los amigos que han estado presente mientras he hecho el proyecto allí en Mariehamn: Erik Hemming (toda una filosofía de vida), Karna, Corinna y toda su adorable familia, a Eric Castro, pero sobre todo a mis compañeros de trabajo, de piso, vecinos, colegas y amigos Juan Carlos (Padre) y Alejandro Alonso. Ellos han convertido esta experiencia en Finlandia en una de las mejores de mi vida con diferencia y hablando de ellos quiero matizar el apoyo profesional que ha sido Alex para mí.

No encontraré ni en un millón de años las palabras para agradecerle todo lo que ha hecho por mí cuando me faltaban las fuerzas, tanto voluntariamente como involuntariamente con esa actitud de oro que tiene, porque ha sido quien me ha tirado una cuerda siempre que me he visto sólo en el pozo en el que a veces he estado durante la realización de este proyecto.

En último lugar quiero mencionar a Miguel Ángel Zurdo, el amable y desinteresado gurú de este proyecto, la única persona que nos ha sacado de dudas en cuanto a tecnología y consejo se refiere.

Gracias a todos ya no tanto por apoyarme durante la realización de este proyecto, sino por formar una gran parte de mí que al ser pegada ha hecho lo que soy ahora mismo, una persona que cree firmemente que el futuro pertenece a quienes creen en la belleza de sus sueños.

Vicente García Adánez.

"Hoy sólo estamos en manos de la naturaleza.
Somos seres humanos, no dejemos que mañana
estemos también en las de nuestras propias creaciones"

V.G.A.

Esta página ha sido dejada en blanco intencionadamente.

HOJA DE ESTADOS.

El presente documento de Hoja de Estado proporciona una historia de problemas y revisiones del documento con un comentario indicando el motivo de la revisión.

Entorno de monitorización de sistemas informáticos embarcados mediante pantallas táctiles			
Informe			
Número	**Revisión**	**Fecha**	**Razón del cambio**
1	0	23-12-2010	Primera versión del documento
2	0	03-01-2011	Presentación al coordinador de la memoria.

TABLA DE CONTENIDO

LISTA DE TABLAS.

LISTA DE IMÁGENES.

LISTA DE CÓDIGO.

SECCIÓN 1: PROLOGO

1. PROPÓSITO DEL DOCUMENTO.

El objetivo de este documento es ser la memoria del Proyecto de Fin de Carrera cuyo desarrollo se considera necesario en el Plan de Estudios para Ingeniería técnica en Informática de gestión de acuerdo con la B.O.E. 27.07.94 como un requisito obligatorio.

Una vez se lleve a cabo la defensa se podrá tener derecho al título de ingeniero técnico en informática de gestión.

2. ACRÓNIMOS Y ABREVIATURAS.

Esta sección proporciona las definiciones de todos los términos, siglas y abreviaturas o se refiere a otros documentos donde puede ser encontrada la definición.

2.1. ACRÓNIMOS.

Acrónimo	Significado
2D y 3D	2 dimensiones y 3 dimensiones
API	Application Programming Interface
BCL	Basic Class Library
CD	Compact Disc
CIL	Common Intermediate Language
CLR	Common language runtime
CLS	Common Language Specification
CPU	Central Processor Uni
CTS	Common Type System
DLL	Dynamic Link Library
GAC	Global Assembly Cache
GDI	Graphics Device Interface
GPU	Graphic Processor Unit
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HTML	HyperText Markup Language
IBM	International Business Machine
JIT	Just in time
MNC	Mobile Net Control
MSDN	Microsoft Developer Network
MSIL	Microsoft Intermediate Language
PFC	Proyecto de fin de carrera
RAM	Random Access Memory
RSA	Rivest, Shamir and Adleman

SDK	Software Deployment Kit
SQL	Structured Query Language
SVG	Scalable vector graphics
UML	Unified Modeling Language
UTF-8	Unicode Transformation Format 8 bit
WCF	Windows Communication Foundation
WCS	Web Coverage Service
WF	Windows Workflow Foundation
WPF	Windows presentation foundation
WPF/E	Windows Presentation Foundation Embedded
XAML	eXtensible Application Markup Language
XML	eXtensible Markup Language

2.2. ABREVIATURAS.

Abreviatura	**Significado**
Ab.	Enterprise
C#	C sharp
C#.NET	C#
C/	Calle
etc.	etcétera
Ing.	Ingeniero
I.V.A	Impuesto de valor añadido
Me.	Millenium
Xp.	Experience
R.R.H.H.	Recursos humanos

3. REFERENCIAS.

Esta sección proporciona una lista completa de todos los documentos a los que el texto de esta memoria se refiere en algún momento, identificado por título, autor y fecha en su caso.

Referencias principales

About.com. (s.f.). *About.com*. Obtenido de Sitio web de About.com, parte de New York Times Company: http://cplus.about.com/od/glossar1/g/frameworkdefn.htm

Blogspot.com. (s.f.). Obtenido de http://3.bp.blogspot.com/_gWQaU40PH24/THat12V_DkI/AAAAAAAAI44/yS8mqo8TILg/s320/expression_blend_f005d908-0287-4a20-95cd-8ff0968205a2.jpg

C.Nagel, B. J. (2008). *Professional C#.* Wiley Publishing, Inc.

codehaus.org. (s.f.). Obtenido de http://docs.codehaus.org/download/attachments/6096/Overview_of_the_Common_Language_Infrastructure.png?version=1&modificationDate=1202485140923

Congci. (s.f.). Obtenido de http://www.congci.com/upload/image/small/20091213004033.GIF

C-Sharpcorner. (s.f.). Obtenido de http://www.c-sharpcorner.com/UploadFile/Dada%20Kalander/MigratingASPtoASP.NET11262005035157AM/Images/01.GIF

Desarrolloweb.C#. (s.f.). *Desarrolloweb.com*. Obtenido de Web para desarrolladores: http://www.desarrolloweb.com/articulos/561.php

Desarrolloweb.SQL. (s.f.). *Desarrolloweb.com*. Obtenido de http://www.desarrolloweb.com/articulos/262.php

Desarrolloweb.VisualStudio. (s.f.). *Desarrolloweb.com*. Obtenido de http://www.desarrolloweb.com/articulos/561.php

Desarrolloweb.XML. (s.f.). *Desarrolloweb.com*. Obtenido de http://www.desarrolloweb.com/articulos/449.php

elChelo. (s.f.). *elChelo wordpress*. Obtenido de http://elchelo.wordpress.com/expression-blend/

Expression Microsoft. (s.f.). *Sitio web de Microsoft Expression*. Obtenido de http://www.microsoft.com/expression/

Finplanet.ru. (s.f.). Obtenido de http://www.finplanet.ru/company/2009/microsoft/windows_3.0.gif

FreeSoftware30. (s.f.). *FreeSoftware30*. Obtenido de http://www.freesoftware30.com/uploads/2009/3/20096251142556277801.jpg

interfacesymposia.org. (s.f.). Obtenido de http://www.interfacesymposia.org/interface/history.html

Jett Fergusson, B. P. (2003). *La biblia de C#.* Anaya.

JohnSmithOnWPF. (s.f.). Obtenido de http://joshsmithonwpf.wordpress.com/2007/09/05/wpf-vs-windows-forms/

Malavida.Blend. (s.f.). *Malavida.com*. Obtenido de http://imag.malavida.com/mvimgbig/download/microsoft-expression-blend-4430-1.jpg

malavida.com. (s.f.). *malavida.com*. Obtenido de http://imag.malavida.com/mvimgbig/download/visual-studio-2002-sp2-137-1.jpg

Microsoft.Office. (s.f.). *Microsoft Office webpage*. Obtenido de http://office.microsoft.com/en-us/visio/visio-2010-features-and-benefits-HA101631752.aspx

MNC Marine Web Page. (s.f.). *MNC Marine*. Obtenido de http://www.mnc-marine.ax/

Mobile Net Control Web Page. (s.f.). *Mobile Net Control Web Page*. Obtenido de http://www.mobilenetcontrol.com

MSDN Microsoft. (s.f.). *Microsoft Developer Network*. Obtenido de Sitio web de MSDN para .NET: http://msdn.microsoft.com/es-es/netframework/default.aspx

MSDN, Blogs. (s.f.). *Blogs MSDN Microsoft*. Obtenido de http://blogs.msdn.com/blogfiles/jasonz/WindowsLiveWriter/AnewlookforVisualStudio2010_CF66/DvX_ShellBase_2.png

Nathan, A. (2006). *Windows Presentation Foundation Unleashed.* Sams.

Seco, J. A. (s.f.). *Desarrolloweb.com, portal para desarrolladores*. Obtenido de Desarrolloweb.com: http://www.desarrolloweb.com/articulos/592.php

SQLcourse.com. (s.f.). *QuinStreet Inc.* Obtenido de http://www.sqlcourse.com/intro.html

Sqlite. (s.f.). *SQLite.org applications*. Obtenido de http://www.sqlite.org/about.html

Studio, M. V. (s.f.). *MSDN Microsoft*. Obtenido de http://msdn.microsoft.com/es-es/vstudio/default.aspx

testech-elect. (s.f.). *testech-elect.com*. Obtenido de http://www.testech-elect.com/ontime/vstudio.gif

TomsHardware. (s.f.). *TomsHardware.com*. Obtenido de http://img.tomshardware.com/us/2008/02/29/cebit_2008/cebit_2008_preview___visual_studio_2008_ide.png

Wikipedia . (s.f.). *Wikipedia*. Obtenido de http://es.wikipedia.org/wiki/Framework

Wpdevelopers. (s.f.). Obtenido de http://wpdevelopers.org/wp-content/uploads/2010/05/xBlend1.png

XAML, M. (s.f.). *MSDN Microsoft*. Obtenido de http://msdn.microsoft.com/es-es/library/ms752059.aspx

4. VISTA PREVIA DEL DOCUMENTO.

Este documento está dividido en 6 secciones organizadas de la siguiente manera:

I. Prólogo. Contiene el propósito general de este proyecto e incluye una lista de acrónimos y abreviaturas que se usarán a lo largo del documento así como la bibliografía referida. Consta de 4 capítulos.

II. Introducción. En esta sección se introduce al lector a este proyecto explicando el alcance y objetivos de la aplicación, la empresa para la que se está desarrollando así como el estado del arte de las tecnologías y herramientas que se usarán. Consta de 5 capítulos.

III. Especificación de requisitos. Contiene la explicación detallada de los requisitos necesarios para crear las diferentes partes que componen este proyecto. Además muestra las soluciones parciales que se han ido dando a lo largo del proyecto para poder seguir la evolución de éste. Consta de 4 capítulos.

IV. Detalles de implementación. Dividida en las tres componentes principales que han formado este proyecto (Plugin, aplicación para la pantalla táctil y base de datos), esta sección contiene una explicación detallada de cómo han sido desarrolladas y cómo funcionan. Consta de 3 capítulos.

V. Planificación de proyecto. Contiene todo lo referente a cómo se ha desarrollado el proyecto en términos temporales y de coste, así como las diferencias entre lo que se estimó que iba a ser de este proyecto y lo que realmente ha sido. Consta de 5 capítulos.

VI. Epílogo. Esta sección es la encargada de presentar las conclusiones del proyecto y sus futuras líneas de trabajo. Además contendrá las instrucciones para ejecutar el software y el contenido del paquete que se proporcionará. Consta de 4 capítulos.

SECCIÓN II: INTRODUCCIÓN

1. INTRODUCCIÓN Y "BACKGROUND"

Este proyecto que he titulado "Entorno de monitorización de sistemas informáticos embarcados mediante pantallas táctiles" ha sido elaborado desde Febrero de 2010 hasta Diciembre del mismo año.

Antes de ser asignado este proyecto para el que hoy estoy escribiendo esta memoria, antes incluso de viajar a Finlandia, se me fue asignado otro por el mismo tutor, Juan Llorens Morillo, para crear un comparador de diagramas de UML en lenguaje C# pero que pasó a rechazarse automáticamente al ser ofrecido el actual, sobre el que me centro íntegramente a excepción de un breve inciso en la primera sección de esta memoria.

Gran parte de este proyecto se ha diseñado en Mariehamn, donde se me concedió una beca Erasmus para el segundo cuatrimestre del mismo año con la universidad Carlos III de Madrid en colaboración con Högskolan På Åland, la universidad destino en Mariehamn, Finlandia, precisamente para la consecución del proyecto de fin de carrera (PFC).

Pasado por tanto un mes desde mi llegada en enero a Mariehamn, se me ofreció la posibilidad de cambiar de proyecto y pasar así a trabajar colaborando para una empresa Finlandesa, MNC, con sede en la propia isla donde estaba viviendo. Sin duda lo acepté, personalmente me llamaba mucho la atención poder trabajar para una empresa real desarrollando un software que en base implantarán en los barcos que trabajen con ellos. Profesionalmente tampoco había duda, aprender cómo se trabaja en una empresa escandinava, con la buena fama que les precede constituía un nuevo desafío y por tanto acepté encantado a hacerlo, no sin tener en cuenta las grandes diferencias con respecto al proyecto anterior y las dificultades que éste entrañaba. Fue un reto para mí.

Durante la estancia de seis meses en Finlandia pudimos reunirnos con el director de MNC en su oficina de Mariehamn, Patrick Sjöberg, y a través de internet mediante conferencias con el responsable de tecnología Miguel Ángel Zurdo, quien se encontraba en España. De este modo se definieron los requisitos y se sentaron las bases técnicas del contenido de esta memoria. Paralelamente MNC abrió para su sección de servicios en barcos además otros proyectos con mis compañeros de beca allí en Finlandia y con los que he trabajado cercanamente:

- Alejandro Alonso Herrera. Su proyecto consiste en realizar un servicio web al que conectarse desde la oficina de Mariehamn y así conocer el estado de los ordenadores de un barco. Su trabajo ha sido de relevada importancia puesto que ha estado ligado directamente al mío teniendo partes en común como es la obtención de información de los componentes informáticos en el barco. El trabajo con él ha sido completamente satisfactorio, se ha dado una colaboración total y una comunicación fluida.

- Juan Carlos Carcelén Fernández. Su proyecto consiste en realizar un disco de arranque para restaurar completamente el sistema de un ordenador embarcado así como de sus datos sin más ayuda que la inserción del propio CD en el ordenador y la conexión de éste a la red.

Tras expirar el periodo de la beca Erasmus y tener el proyecto inacabado regresé a Madrid donde continué el proyecto hasta la consecución del mismo.

1.1. PROYECTO ASIGNADO PREVIAMENTE.

En este apartado y sólo a modo de introducción voy a describir brevemente el proyecto que se me asignó antes del actual, en el que se centra esta memoria. Ambos proyectos no tienen apenas relación pero he creído conveniente hablar de él porque su base teórica y tecnológica me ayudó a plantear y consolidar las bases de éste actual. Supuso la introducción al marco de trabajo .NET, así como al lenguaje C#.

Con la asignación de la beca Erasmus 2010 en Mariehamn llegó la asignación del tutor del proyecto, Juan Llorens Morillo, del departamento de ingeniería del software de la universidad Carlos III de Madrid.

Antes de irme a Finlandia quise tener un proyecto asignado para poder llevar parte de la bibliografía en español e ir avanzando e informándome sobre él. En diciembre me ofreció desarrollar un comparador de diagramas UML.

La idea era crear una aplicación usando el framework o marco de trabajo .NET y su lenguaje por excelencia como lo definen algunos, C#, que tuviera como entrada al sistema dos modelos UML. Su cometido sería obtener y analizar las diferencias entre ambos para luego ser mostradas a modo de resumen con la mayor claridad posible ya no sólo como impreso, sino a través de una página web que se habilitaría posteriormente.

Con el fin de mostrar al lector de una manera clara la introducción a este proyecto, que no debemos olvidar, quedó obsoleto cuando se me asignó el actual, voy a usar el ejemplo de lo que haría el programa en el caso de comparar dos casas. Obviamente en un caso real cada casa tendría muchas más características, pero incluiré sólo unas pocas simplemente para que quede claro su funcionamiento. Este es el modelo de datos de cada casa.

- **Nombre de la casa**. Nombre que se le asigna a una casa. Un ejemplo sería: “Casa de vacaciones Familia García”.
- **Dirección de la casa**. Emplazamiento físico de ésta. Una dirección válida sería “C/Herrén Larga número 8”.
- **Superficie.** Medida en metros cuadrados.
- **Habitaciones**. Cada una de las que componen la casa.
- **Puntos de luz**.

El diagrama de clases UML realizado con Microsoft Visio 2007 correspondiente a las características indicadas sería el siguiente:

Ilustración 1: Diagrama de clases proyecto antiguo

Mediante estos diagramas podríamos representar las dos casas variando los valores de los atributos y la multiplicidad de cada una de las clases.

La aplicación tendría que generar un archivo de marcado XML a partir de las características de cada casa, resumiéndolas y siendo el archivo con el que se trabajará en un futuro. La aplicación a desarrollar tendría que ser capaz de detectar los objetos de los dos archivos XML correspondientes a las dos casas, detectar las diferencias y finalmente:

- Mostrar un informe con las diferencias que hay entre las dos. Por ejemplo: La casa 1, con dirección en C/Herrén Larga 8, de 123 metros cuadrados y 12 puntos de luz, tiene 22 metros cuadrados y 3 puntos de luz más que la segunda casa, con dirección en la C/ Ramón Méndez 21.
- Guardar este informe en una base de datos accesible desde una web.

De cada diagrama en el que se representa una casa podemos obtener un archivo XML que tendría este formato:

```
<Casa>
        <Nombre />
        <Dirección />
        <Superficie />
        <Habitación>
                <Nombre/>
                <Numero/>
        </Habitación>
        <Puntos de luz/>
</Casa>
```

Fragmento de código 1. Formato de diagrama XML para una casa

Donde aplicado al primer ejemplo dado en esta memoria de casa se vería de esta manera:

```
<Casa>
        <Nombre Casa de vacaciones de la familia García/>
        <Dirección C/Herren larga 8/>
        <Superficie 123/>
        <Habitación>
                <Nombre CuartoBaño/>
                <Numero 2/>
        </Habitación>
        <Habitación>
                <Nombre Cocina/>
                <Numero 1/>
        </Habitación>
        <Puntos de luz 12/>
</Casa>
```

Fragmento de código 2. Casa 1 en XML

```
<Casa>
        <Nombre Residencia habitual de Juan Pérez/>
        <Dirección C/Ramón Méndez 21/>
        <Superficie 101/>
        <Habitación>
                <Nombre CuartoBaño/>
                <Numero 1/>
        </Habitación>
        <Habitación>
                <Nombre Cocina/>
                <Numero 1/>
        </Habitación>
        <Puntos de luz 9/>
</Casa>
```

Fragmento de código 3. Casa 2 en XML

Finalmente esta sería la información con la que trabajaría el programa y por tanto, de la que obtendrá las conclusiones y resúmenes.

Nombre	Casa 1	Casa 2
Dirección	C/ Herrén larga 8	C/ Ramón Méndez 21
Superficie	123 metros cuadrados	101 metros cuadrados
Puntos de luz	12	9
Cuarto de baño	2	1
Cocina	1	1

Tabla 1. Resumen de la información comparada de dos casas.

INFORME FINAL DADOS LOS PARÁMETROS:

- La casa 1 tiene 22 metros cuadrados habitables más que la casa 2, lo que significa que es más grande pero gastará más en calefacción, limpieza, etc.
- La casa 1 tiene un cuarto de baño más que la casa dos. Ganará espacio pero gastará más agua, entre otras consecuencias.
- La casa 2 tiene 3 puntos de luz menos que la casa 1.

El ámbito de aplicación de este software no sólo sería el inmobiliario sino que podría usarse en cualquier otro mientras sus objetos puedan ser representados mediante un diagrama UML, o sea, todos de una manera u otra.

El objetivo sería esclarecer las dudas de un usuario y presentarle las diferencias entre los dos modelos de manera clara, por ejemplo a la hora de comprarse un coche, elegir un viaje, una casa… Pudiendo surtirse de los resultados mediante un informe directo o simplemente accediendo desde su propia casa a una página web con una sección de cliente donde ver los resultados gracias a la base de datos implementada.

Uno de los requisitos principales era trabajar con la plataforma .NET y con el lenguaje de programación C#. Para ello, durante el primer mes de estancia en Finlandia me dediqué a recopilar información acerca de estos y a introducirme en dichas tecnologías, valiéndome el conocimiento para el desarrollo del proyecto que se me asignaría más tarde en colaboración con la empresa MNC: “Entorno de monitorización de sistemas informáticos embarcados mediante pantallas táctiles”, y sobre el cual se va a centrar íntegramente el resto de esta memoria.

1.2. EL PROYECTO ACTUAL.

En el primer mes de mi estancia en Mariehamn me dediqué a recopilar bibliografía y a aprender a usar C#.NET así como a refrescar la información previa que tenía sobre UML, SQL… hasta que se me ofreció la oportunidad de hacer este proyecto para el que hoy escribo la memoria.

El actual, según me explicó mi coordinador en un principio, trataba sobre desarrollar una aplicación diseñada para pantallas táctiles colaborando con MNC, una empresa Finlandesa con sede en la ciudad donde vivía, Mariehamn, a las órdenes de su director Patrick Sjöberg. A través de la aplicación creada se debería poder acceder a la información de los diferentes ordenadores que se encontraban en el barco y a otras funciones. Se me adjuntó una breve especificación de requisitos y tras estudiarla decidí aceptarlo, me gustaba la idea de poder trabajar con una empresa real, adquirir conocimientos profesionales que no habría podido tener con el proyecto anterior y aprender a moverme por esos entornos. Además me encantaba la idea de poder trabajar con una empresa Finlandesa debido a la buena fama que les precede.

El tiempo que había dedicado al anterior proyecto no fue en vano puesto que las tecnologías que iba a usar en éste como .NET y los lenguajes C# o SQL eran los mismos. Incluso formarme más en UML, a pesar de que no lo he usado mucho en este proyecto, me ha ayudado a tener más claras las ideas debido a la capacidad de abstracción que implica conocer este lenguaje.

Del mismo modo que a mí se les ofreció a mis dos compañeros de beca Erasmus Juan Carlos Carcelén y Alejandro Alonso la posibilidad de hablar con Patrick Sjöberg y que les asignara otro proyecto a cada uno en colaboración con MNC.

1.3. ALCANCE, DOMINIO Y OBJETIVOS DEL PROYECTO.

1.3.1. ALCANCE

El trabajo a realizar es el desarrollo de una aplicación software que funcione en una pantalla táctil que se supondrá en el puente de mando de un barco y que tendrá dicha aplicación abierta permanentemente para siempre monitorizar el estado de todos los ordenadores, entre otras funciones.

Al terminarse el proyecto entero, lo que se entregará será lo siguiente:

- La aplicación desarrollada.
- La infraestructura necesaria para poder trabajar con ella.
- Amplia memoria del proyecto (varias copias).

Este proyecto ha sido diseñado para funcionar con un total de cinco ordenadores que habría que conectar. Este requisito ha sido definido puesto que cinco son los ordenadores de los que constará el paquete básico que MNC comercializa.

Para la realización de este proyecto ha sido necesario tener conocimiento de:

- **La plataforma Microsoft .NET.** Siendo uno de los requisitos principales, toda la aplicación será desarrollada bajo este marco de trabajo y por tanto es importante conocer sus características así como sus tecnologías para poder hacer una elección óptima de las herramientas que nos brinda.
- **Manejo del lenguaje de programación C#.NET.** Es un requisito primordial puesto que es el lenguaje con el que MNC ha desarrollado la mayoría de sus aplicaciones.
- **Manejo de interfaces con Windows Forms o WPF.** Toda aplicación necesita una interfaz llamativa y mucho más si se trata de una aplicación para pantalla táctil. MNC creaba hasta ahora sus interfaces usando Windows Forms, pero dado que esto se está quedando obsoleto se propusieron gracias a Miguel Ángel Zurdo, dar el salto a los proyectos WPF. Por eso voy a desarrollar la primera interfaz usando dicha tecnología.
- **Manejo de bases de datos y del lenguaje SQL**. La información que se vaya recolectando de los ordenadores se almacenará en ellas. Se utilizará SQLite para trabajarlas.
- **Gestión de proyectos**. Este proyecto comenzado desde cero tiene que ser pensado y planificado con detalle para la consecución de sus metas. Lo único que se me ha dado es una especificación de requisitos. Todo lo demás, desde planificaciones a personal necesario, ha sido ideado y desarrollado en exclusiva por la misma persona.
- **La empresa MNC.** Es necesario conocer la empresa y sus productos. El paquete básico que MNC ofrece es de cinco ordenadores, circunstancia que he aprovechado para definir el número de elementos a monitorizar en un barco. De este cometido se ha encargado Miguel Ángel Zurdo, quien se ha encargado de explicar el funcionamiento del sistema de MNC y quien ha resuelto las dudas surgidas.

A modo de resumen, la funcionalidad del sistema puede dividirse en las siguientes partes:

1. Creación de un archivo XML con el estado de los elementos hardware del ordenador que hayamos decidido monitorizar, gracias al software instalado en cada uno de los ordenadores del barco que tengan instalada la aplicación pertinente.

2. Envío de dichos archivos al servidor en el barco.

3. Inserción de su información en la base de datos local del barco.

4. Lectura de la base de datos para poder mostrarlos por pantalla con la aplicación para pantalla táctil.

1.3.2. DOMINIO

Este proyecto, cuyo entregable será la aplicación mencionada junto con los complementos necesarios para su correcto funcionamiento será usado por la empresa MNC sobre barcos, sin especificar tipo ni propósito de estos. Cualquier empresa o particular que esté interesado en contratar los servicios de MNC podrá tener acceso a este software, ya que igual que se puede usar para barcos puede usarse para oficinas u otros entornos de trabajo.

Actualmente sabemos que MNC tiene clientes con diferentes modelos de barco, desde industriales con más ordenadores a otros yates privados con apenas unos pocos.

1.3.3. OBJETIVOS

EL objetivo principal de este proyecto es desarrollar una aplicación software que funcione en una pantalla táctil localizada en el puente de mando de un barco y que la tendrá abierta permanentemente.

Será fácil e intuitiva como para que pueda ser manejada por gente sin formación previa y servirá para tener un claro esquema de los componentes informáticos que hay en un barco así como del estado en el que se encuentran.

El desarrollo de este proyecto pretende introducir a MNC de lleno en el mercado naval, más en concreto en el mercado informático naval, constituyendo una aplicación que debería ser básica para cualquier barco y que predomine por encima de sus competidores gracias a su simpleza y buen rendimiento.

1.4. FUNCIONAMIENTO GENERAL DEL PROYECTO.

En esta sección quiero explicar a modo de introducción cómo funciona todo este sistema que he creado para este proyecto.

Al estar compuesta de varios elementos, he considerado apropiado en este apéndice introductorio explicar cómo interaccionan entre ellos para que el lector, antes de comenzar a leerse el cuerpo de la memoria tenga una idea gráfica en su cabeza que le haga entender este sistema, qué flujo de datos se dan entre unos y otros elementos, secuencias de ejecución, etc.

El propósito de la detallada estructura de esta memoria es aclarar al lector; entiendo que leer los distintos apartados de esta memoria puede resultar abstracto si no se sabe el por qué de su papel en mi proyecto. Por eso he creído conveniente crear primero un esquema de funcionamiento de mi aplicación y comentar los detalles que a ello conciernen. De este modo al leer partes como por ejemplo "el estado del arte" podrá conocer para qué se usan dentro de mi sistema y así entender mucho mejor todos los conocimientos teóricos y prácticos que aquí reflejo ya que es necesario una visión global de todos los elementos de mi sistema como uno propio, no como elementos por separado.

Lo primero que quiero que el lector tenga claro es que siempre se ha trabajado bajo el Framework .NET 3.5 y que los siguientes son los elementos que participan en este proyecto:

- **El programa de MNC "Mobile Net Control 2.0".** Es el software que proporciona la empresa para controlar los ordenadores que proporcionan a los clientes. Contiene una serie de plugins o pequeñas aplicaciones que le aportan diferentes funcionalidades. Relativo a mi proyecto, uno de estos es el "Detailed Info Plugin" que se ha creado exclusivamente para este proyecto y que explico a continuación.

- **El plugin "Detailed Info"**. Es uno de esos pequeños programas o funcionalidades que funcionan en el programa "Mobile Net Control 2.0" y que hemos creado usando el Framework .NET y su lenguaje de programación C#. Es simplemente un código sin interfaz que generará el archivo .dll (que será el plugin en cuestión) con la función necesaria para detectar la información del sistema donde está funcionando, generar un informe mediante un archivo XML, enviarlo al servidor local y que éste introduzca dicha información en una base de datos también ubicada en el mismo servidor. El usuario no tendrá pruebas fehacientes a tiempo real de que esta operación ha sido exitosa, más que la propia inserción de los datos en la base de datos puesto que se trabaja sin emitir notificaciones. De cara a la presentación de este proyecto se han incluido dos notificaciones para aclarar cuándo está el plugin trabajando.

- **La aplicación para pantalla táctil "Touchable Screen Monitor Application"**. Es la parte de la aplicación que contiene el código y la interfaz con la que el usuario con los elementos informáticos del barco así como con la base de datos y su información. Tanto la funcionalidad como la interfaz están integrados bajo un mismo proyecto WPF de Microsoft .NET. Para cada una de las ventanas que se han diseñado se generan dos archivos con diferentes propósitos:
    - Aspecto de la ventana: Contiene simplemente las cajas, los colores, los marcos, botones… que compondrán el aspecto visual de la interfaz. Se puede decir que es el escenario donde tiene lugar la obra de teatro que es la funcionalidad de la aplicación. Está implementada en lenguaje XAML mediante las herramientas

Microsoft Expression Blend 3 y Microsoft Visual Studio 2008. El archivo generado con tal propósito es aquel con extensión .xaml.

- La propia funcionalidad de la interfaz: Implementada en C#, esta parte contiene los eventos que se lanzarán cuando pulsemos un botón en el modelo de pantallas. De este modo al pulsarlo en esa parte se acudirá a esta otra donde se ejecutará el código pertinente y que generará el cambio de ventana o de los propios elementos que la componen, entre otros. Dentro de cada ventana supone el archivo con extensión .cs.

- **La base de datos.** Constituye un elemento ligado a las dos últimas partes que en este capítulo acabo de definir. Es una base de datos creada con SQLite y que he administrado con el programa SQLite administrator, de distribución gratuita y compatible con el código implementado previamente en los programas de MNC.
  Su primera función es almacenar los datos de los ordenadores que el plugin de MNC está mandando al servidor local del barco y que éste está insertando, de este modo los tendremos todos ordenador. Más tarde será la aplicación "Touchable Screen Monitor Application" quien acceda a ella para leer estos datos y poder mostrarlos por pantalla.

- **El servidor local.** Es un servidor que se supone en el propio barco donde se está trabajando. Para recrearlo en los ordenadores de pruebas, existe instalado en un directorio distinto al del programa principal MobileNetControl 2.0, un servicio llamado MobileNetControl Repository Management Service que habilitará este servidor virtual, de modo que cuando éste esté iniciado, el programa Mobile Net Control 2.0 podrá interactuar con él. De este modo se le podrá mandar la información necesaria para el correcto funcionamiento de las aplicaciones. Se accederá a él por el puerto 5054.

Este es el esquema de funcionamiento de mi aplicación para que el usuario lo tenga siempre presente de ahora en adelante:

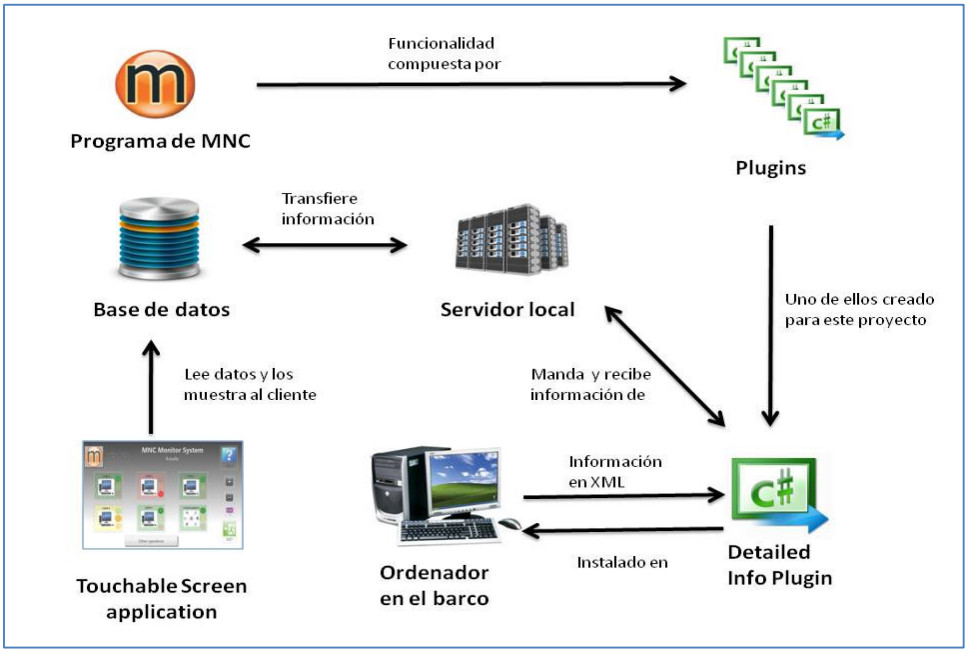


Ilustración 2. Funcionamiento general de la aplicación.

Para aclarar la situación física de cada elemento quiero explicar unos conceptos que quizás no hayan quedado del todo claros. Todos los elementos mostrados en el esquema se encuentran en el barco. En el caso de que se quiera acceder a ellos desde fuera, como por ejemplo desde las oficinas de MNC en Finlandia, se hará a través del servidor utilizando la tecnología remoting, proyecto del que se encarga mi compañero Alejandro Alonso Herrera.

Por tanto, dentro del barco y del esquema mostrado podemos diferenciar claramente tres elementos claves:

- Ordenadores a monitorizar. Contendrán el siguiente software.
    - **Programa de MNC "Mobile Net Control 2.0".** Con una serie de plugins.
    - **Detailed Info Plugin.** Desarrollado en este proyecto para el programa de MNC.

- Pantalla táctil en el puente de mando del barco. Contendrá:
    - **Touchable Screen Monitor Application.** Desarrollada en este proyecto, servirá para interactuar con el usuario.

- Servidor local. Contendrá los siguientes elementos.
    - **Base de datos** con la información almacenada de cada ordenador.

2. LA EMPRESA MOBILE NET CONTROL ESCANDINAVIA (MNC)

2.1. INTRODUCCIÓN A LA EMPRESA MNC.

Ilustración 3. Logo de MNC.

Nombre de la empresa: MobileNetControl Scandinavia Ab.

Dirección: Östra Esplanadgatan 7, 22100, Mariehamn, Åland, Finlandia.

Número de contacto: +358 (0)18 22773 (Patrick Sjöberg, director)

Dirección de contacto: http://www.mobilenetcontrol.com/contact.aspx

Dirección de la página web: www.mobilenetcontrol.com (actualmente esta página está disponible por defecto en idioma inglés pero también se ofrece en Español y en Sueco).

Mobile Net Control o MNC, como me refiero a ella durante toda la memoria para abreviar, es una empresa nacida en el año 2006 con sede en Mariehamn, capital de las islas Åland, provincia de Finlandia.

Se dedican a ofrecer soluciones y servicios relacionados con el campo de la informática para empresas o particulares. Más concretamente en el de la seguridad en ordenadores, dando soporte y creando copias o "backups" de la información. Para este cometido han desarrollado un novedoso procedimiento con el cual un cliente al contratar a MNC para que se encargue de la seguridad de su ordenador, comprima y codifique sus archivos de manera única para su almacenaje en servidores de MNC donde los clientes pueden tener sus datos de manera segura y acceder a ellos en caso de posibles pérdidas o problemas en los archivos de sus ordenadores base.

Una de los propósitos de MNC es proveer a los barcos con sistemas de ordenadores fiables asegurando que cualquiera de sus equipos será plenamente recuperado en menos de una hora con sus procedimientos.

MNC tiene colaboradores en países como España, Suecia, Finlandia o Perú y sus inversores principales son:

- Ålands Ömsesidiga Försäkringsbolad.
- Ålands Utvecklings Ab.
- Inversiones para la tecnología de la información.
- Otros particulares.

2.2. SERVICIOS QUE OFRECE MNC

1. Asegura tu ordenador.

Asegura la información y los programas de su ordenador contratando este servicio.

No importa si es personal o de empresa, MNC asegura que si existe algún problema podrá recuperar tanto la información personal como el estado de su sistema operativo, particiones, programas… a un momento anterior al problema. Todo esto con sólo hacer un "Click" o pulsar una tecla, los programas instalados se encargarán del resto avisándole cuando el proceso haya finalizado.

Si el problema es configurar un ordenador, como podría ser el caso de una restauración, lo único que tiene que hacer es introducir un disco de arranque que MNC le proporcionará y escribiendo la poca información necesaria, el CD instalará su ordenador al completo sin necesidad de saber nada de informática.

Este producto está orientado a ahorrarle tiempo al usuario. Tanto hacer backups como restaurar un sistema lleva una gran cantidad de tiempo pero con los programas de MNC se hará de forma automática ahorrándole una gran cantidad de tiempo al usuario y asegurando que los sistemas tendrán todo lo que el usuario quiere tener debido a su innovador método. (Mobile Net Control Web Page)

Los paquetes de servicios que se ofrecen son los siguientes:

- Asegura un ordenador de empresa. Hace backups de forma automática pasado un determinado periodo de tiempo. La información siempre estará encriptada para garantizar la confidencialidad de los datos. Este servicio ofrece soporte telefónico, vía e-mail o tomando el control de su ordenador desde las oficinas de MNC.

  ¿Cómo funciona? Guarda el último backup de cada mes durante los últimos 12 meses, de forma que si se quiere volver a un estado anterior se pueda elegir.

  El precio es de 138 euros al año (11.50 al mes) sin incluir tasas.

- Asegure un ordenador personal. Hace backups de forma automática y la información siempre estará encriptada.

  ¿Cómo funciona? Guarda el último backup de cada mes durante los últimos 4 meses, de modo que se puede regresar a un estado anterior en ese periodo de tiempo.

  El precio es de 120 euros al año (10 euros al mes) tasas incluidas.

2. Asegura tu trabajo.

Guarda codificada toda la información archivo a archivo de su ordenador en el servidor de MNC. Es una buena manera de tener copia de seguridad tanto de archivos críticos como de información que no quiere perder.

El precio depende del volumen de datos que quieras asegurar, son 10 euros por cada paquete de 20 Gigabytes que se contrate.

3. Asegura tu configuración.

Con este servicio puedes hacer un backup de tu ordenador incluyendo todos los programas que tenías y el estado del sistema.

De este modo puedes guardar ya no sólo archivos si no también Configuraciones de red, de periféricos como impresoras, scanner, pantalla y programas que se arrancan al iniciar el sistema operativo entre otros.

4. Asegure sus contraseñas.

Almacene de forma segura todos los nombres de usuario y contraseña de las diferentes páginas que navegue o correos electrónicos, números de tarjetas de crédito, de banco, teléfonos... Gracias a MNC y a este servicio el usuario podrá acceder a toda esta información cuando lo desee y por tanto no se verá obligado a aprenderse todas las contraseñas; sólo tendrá que aprenderse la propia para acceder al servicio de MNC.

Toda la información mostrada en este capítulo ha sido obtenida de (Mobile Net Control Web Page).

2.3. MNC MARINE.

MNC Marine es un área de MNC especializado en los servicios y operaciones orientados a los barcos y a la industria naval.

Dado que MNC tiene su sede en las islas Åland y que tiene una gran actividad marítima, se intenta acceder a este sector vendiendo servicios y productos.

Para ello MNC ha desarrollado un sistema único de mantenimiento y monitorización de equipos y de su información que combina la gran seguridad que esta empresa ofrece con sus métodos, con un bajo coste al alcance incluso de particulares o pequeñas empresas. De este modo el personal del barco, aunque no esté formado en el manejo de ordenadores, puede fácilmente restaurar desde un equipo en su totalidad hasta una selección de archivos que sin querer ha borrado.

¿Qué se pretende con esto? Que sea MNC quien se encargue al 100% de todo lo que ocurre con los componentes informáticos de la embarcación y que por tanto el personal del barco sólo tenga que centrarse en sus tareas.

La empresa tiene 18 años de experiencia trabajando en el mantenimiento de ordenadores y más concretamente 15 en el campo naval. Hoy por hoy MNC mantiene más de 130 sistemas repartidos en diferentes embarcaciones.

Principalmente los servicios que les ofrece son:

- Backup y recuperación de la información de los ordenadores.
- Recuperación ante fallos de software y hardware.
- Monitorización de software y hardware.
- Soporte online, por teléfono o en persona a los barcos.

Hay que tener en cuenta que realmente MNC Marine es el área en la que debo incluir este proyecto puesto que mi aplicación irá designada a una pantalla táctil situada en el puente de mando del barco a la que se asigne.

Servicios que ofrece MNC para barcos.

- Asegura tu barco con MNC: Contratando este servicio tendrás todo el hardware y el software de un barco asegurado las 24 horas del día todos los días del año. Podrás recuperar toda tu información o sustituir ordenadores que se estropeen, por ejemplo en alta mar, en pocos pasos y de manera sencilla gracias a las instrucciones provistas.

  Al llegar a puerto el personal de MNC se encargará de recoger el material defectuoso, arreglarlo o proveer con nuevo al cliente en el barco.

- Asegure su ordenador con MNC: Este servicio proporciona la seguridad de que nunca perderás tu información hagas lo que hagas con ella porque MNC tiene copias almacenadas en sus servidores. Así garantiza que la recuperará sin problema en un periodo de cómo máximo 1 hora, caso que se daría cuando se tiene que recuperar toda la información del ordenador. El tiempo máximo al que podrás remontarte para restaurar tus archivos es de 3 años.

- Soporte efectivo ante problemas: Además de soporte telefónico y por email siempre activo, MNC pone a disposición de sus clientes mediante su página web http://www.mobilenetcontrol.com/onlinesupport.aspx un servicio de NTRGlobal que permite al personal de MNC tomar el control del ordenador esté donde esté siempre que tenga conexión a internet. De este modo MNC solucionará el problema y el mismo cliente podrá aprender para la próxima vez que le ocurra

3. ESTADO DEL ARTE DE LAS TECNOLOGÍAS Y LENGUAJES UTILIZADOS.

En esta sección quiero introducir al lector en las tecnologías y lenguajes que he usado durante el desarrollo de este proyecto. Lo voy a hacer clasificándola en dos diferentes niveles cognoscitivos para los diferentes tipos de lectores que pueden acceder a esta memoria. ¿Para qué? Aunque la mayoría de los lectores serán técnicos profesionales creo que es importante reconocer que puede haber otros tipos de público y por tanto me gustaría de algún modo no aburrirles, que puedan elegir sin tener que leer todo y adquieran de una manera dinámica los conocimientos que quiero inculcar según sus expectativas.

Por tanto, los dos principales niveles en los que dividiré esta sección serán:

Una lista de elementos para que un primer lector pueda reconocer las tecnologías y lenguajes con las que he trabajado. En esta lista no se profundizará aunque sí se explicará brevemente cual ha sido su cometido y su importancia. Estará orientado a personas que ya las conocen y que por lo tanto sólo necesitan saber que las he utilizado.

Niveles detallados. Más allá de la lista, está pensado para la gente que quiere profundizar y aprender más sobre el tema en cuestión. A su vez he creído conveniente crear los siguientes apartados:

> **Introducción**. Destinada a aquel lector que quiera adquirir un conocimiento básico del tema sin llegar a profundizar en las amplias posibilidades y en los detalles de implementación.
>
> **Profundidad**: Este es el nivel más técnico, para aquel lector que además de saber qué he utilizado para este proyecto quiera formarse al respecto de una manera más profunda. Aquí puede encontrar ya no sólo información teórica al respecto sino también práctica y que sirva como un primer nivel de aprendizaje de la tecnología o el lenguaje en cuestión.

3.1. LISTA DE LAS TECNOLOGÍAS Y LENGUAJES USADOS.

Marcos de trabajo y tecnologías

El marco de trabajo es una colección de clases y aplicaciones, librerías de SDKs y APIs, bibliotecas y lenguajes, creado para que todos los componentes puedan trabajar juntos. (About.com)

- Microsoft .NET Framework 3.5. Es el marco de trabajo en el que se ha basado mi proyecto. Casi todo lo que he hecho ha sido sobre él, incluyendo la tecnología WPF y el lenguaje de programación C#, entre otros.
- WPF (Microsoft Windows Presentation Foundation). Es la tecnología que pertenece al marco de trabajo .NET que me ha permitido desarrollar la interfaz de mi aplicación de un modo más innovador puesto que hoy por hoy es considerada por los expertos como la más potente que ofrece Microsoft.
- Microsoft Windows 7. Ha sido el sistema operativo que he utilizado. Sin él no sería posible utilizar el marco de trabajo .NET.

Lenguajes utilizados:

Los lenguajes son aquellos idiomas en los que he escrito el código de mi aplicación.

- C# (C Sharp). Es el lenguaje por excelencia del marco de trabajo .NET. Con él he escrito tanto la aplicación "Touchable Screen Monitor Application" como el plugin "Detailed Info Plugin".
- XML (eXtensible Markup Language). Los modelos de datos con los que trabaja mi aplicación se han basado en estructuras definidas en este lenguaje.
- XAML (eXtensible Application Markup Language). Es el lenguaje con el que está escrita la interfaz de la aplicación "Touchable Screen Monitor Application". Es parecido al XML pero con variaciones.
- SQL (Structured Query Language). Es el lenguaje usado para crear, acceder y operar sobre la bases de datos que he necesitado para este proyecto. He utilizado librerías SQLite.

Herramientas utilizadas:

Estas tecnologías y lenguajes de los que he hablado anteriormente no son posibles en la práctica si no se utilizan una serie de herramientas que trabajen con ellos. Estos son los diferentes programas que conocen sus reglas y características y que nos permiten utilizarlos para nuestros propósitos.

- Microsoft Visual Studio 2008. Es un entorno de desarrollo integrado que me ha permitido crear la aplicación usando los lenguajes C# y XAML mediante una interfaz de ventanas.
- Microsoft Expression Blend 3. Es una herramienta profesional desarrollada por Microsoft que permite crear experiencias de usuario más atractivas y de manera más fácil que con Visual Studio.
- Microsoft Visio Professional 2007. Una herramienta de dibujo vectorial que he usado para realizar los esquemas y diagramas que aparecen en esta memoria.

3.2. NIVELES DETALLADOS DE LAS TECNOLOGÍAS Y LENGUAJES USADOS.

3.2.1. .NET FRAMEWORK 3.5.

Ilustración 4. Logo de Microsoft .NET

INTRODUCCIÓN

La información que concierne a este capítulo ha sido obtenida a raíz del estudio de dos libros como principales fuentes, (C.Nagel, 2008), (Jett Fergusson, 2003) y ha sido complementado con (Seco).

.NET es un conjunto de nuevas tecnologías de Microsoft orientadas a la creación de software para Internet. Se basarán en servicios web con el objetivo de crear un entorno de desarrollo y ejecución de software que pueda ser accedido independientemente del lenguaje de programación, sistema operativo ó hardware que se use. (MSDN Microsoft)

Hay que saber que para poder trabajar con .NET es necesario tener un sistema operativo de Windows y el conjunto de herramientas .NET Framework SDK, descargable de forma gratuita en http://www.msdn.microsoft.com/net. Al descargar este marco de trabajo estaremos también descargando compiladores de los principales lenguajes de la plataforma como C#.NET, Visual Basic.NET o JScript.NET.

Para este proyecto se ha trabajado en dicho entorno y en mi caso no ha sido necesario descargar el framework .NET 3.5 puesto que los ordenadores donde mi proyecto ha sido testado los tenían incluidos en sus diferentes versiones de sistema operativo Windows 7.

A pesar de que originalmente .NET trabajaba con los lenguajes mencionados anteriormente y sólo sobre plataformas Windows, al tener Microsoft que publicar la norma que define el conjunto de funciones más importantes que implementará .NET ha supuesto que también se va a poder trabajar sobre el marco del código abierto y terceros por tanto han creado versiones para integrar en esta plataforma lenguajes como Cobol, APL, Oberon, Java o Phyton entre otros. De esta manera .NET no sólo tendrá los

lenguajes iniciales como C#, JScript o Visual Basic sino que seguirá creciendo y aumentando sus posibilidades en función de su demanda.

El núcleo de .NET: El CLR

Todo el Framework se basa en esta aplicación parecida a una máquina virtual y que gestiona la ejecución de las aplicaciones de .NET. El CLR facilita su desarrollo, mantenimiento y favorece su fiabilidad y seguridad.

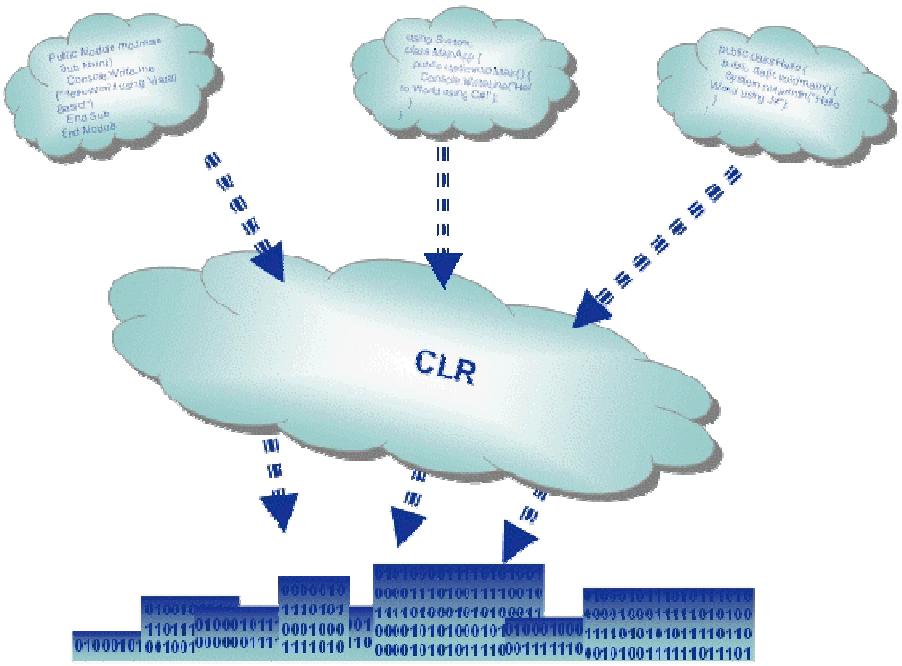


Ilustración 5. Esquema básico del CLR (Congci)

Como puede verse en la imagen, la función básica del CLR es traducir el código del programa que estemos desarrollando escrito en cualquier lenguaje, a código máquina específico de nuestro ordenador.

¿Cómo lo hará? Esta información debido a su complejidad viene explicada en el siguiente apartado ".NET en profundidad".

¿Qué hacer para tener el framework .NET?

Microsoft ha publicado:

- .NET Framework SDK. Es el kit de desarrollo de software que incluye las herramientas necesarias tanto para su desarrollo como para su distribución y ejecución. Es posible descargarlo de manera gratuita desde http://www.msdn.microsoft.com/net.
- Visual Studio.NET, que permite desarrollar proyectos desde una interfaz visual basada en ventanas. Este se descarga de la misma página que .NET Framework SDK pero para hacerlo debes ser suscriptor MSDN universal.

.NET EN PROFUNDIDAD.

Tras la introducción ofrecida en el apartado anterior quiero explicar más en profundidad los pilares del marco de trabajo .NET.

El CLR: "Common Language Runtime" de Microsoft .NET

Como he explicado antes, es el núcleo de la plataforma .NET y el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas. Les ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad.

Como el CLR gestiona código, al de las aplicaciones creadas para la plataforma .NET se le suele llamar código gestionado. Al código no escrito para ser ejecutado directamente en la plataforma .NET se le suele llamar código no gestionado.

A continuación voy a mostrar las 13 principales características que he considerado más importantes para definir al CLR de Microsoft .NET, que son:

1. Ofrecer un modelo de programación orientado a objetos. Esto es una diferencia importante respecto al modo de acceso a los servicios que ofrecen algunos sistemas operativos actuales (entre los que incluyo a los de la familia Windows), donde se accede a los servicios a través de llamadas a funciones globales definidas en DLLs, lo que trae muchos problemas debido a la gran cantidad de versiones de estas.

2. Ofrece un modelo de programación sencillo. Con el CLR desaparecen muchos elementos complejos tales como registros de Windows, GUIDs* o HRESULTs** que se dan en los sistemas operativos actuales.

* GUID (Globally Unique Identifier): Es un tipo de identificador (presentado en hexadecimal de 32 caracteres pero almacenado como entero de 128 bits) que se usa en aplicaciones software para describir un número de referencia único.

** HRESULT: Es un tipo de dato usado en la tecnología de Microsoft como parámetro de funciones y valores retornados para describir errores y "Warnings" en los programas.

3. Eliminación del sistema basado en DLLs. Tanto en Windows, el sistema operativo más común, como en otros menos comunes, se tiene el problema de basar el trabajo en archivos DLL, tal como he comentado en el primer punto de esta sección. Esto hace que al tener diferentes versiones de archivos DLL hubiera problemas de incompatibilidad. En .NET y gracias al CLR todas las versiones pueden coexistir, simplificando mucho el desarrollo de software.

4. Ejecución multiplataforma: El CLR actúa como una máquina virtual que ejecuta las aplicaciones diseñadas para la plataforma .NET. Es decir, que Microsoft ha desarrollado versiones del CLR para la mayoría de las versiones de Windows: 95, 98, ME, NT 4.0, 2000, XP, CE, Vista, 7.

Al ser la arquitectura del CLR de dominio público, Microsoft lo va a portar al sistema operativo Linux aunque paralelamente también existen terceros que están desarrollando de manera independiente otras versiones del mismo. No se descarta que se desarrollen versiones para otros sistemas operativos en el futuro.

5. Integración de lenguajes. Es posible usar código de cualquier lenguaje si para éste existe un compilador que genere el código necesario para la plataforma .NET.

De este modo se han creado compiladores para lenguajes como C#, Visual Basic, C++, JScript y otros que están en camino desarrollados por Microsoft o por terceros como he explicado anteriormente.

Así se facilita mucho el trabajo puesto que hay partes que un desarrollador puede querer hacerlas en otro lenguaje, abriendo un mundo de posibilidades en este sentido.

6. Eficiencia en la Gestión de memoria gracias al recolector de basura. Esta es una de las principales características que incluye el CLR. Al hablar de basura se refiere a los objetos que se han creado en la ejecución de nuestro programa y que no se van a usar de ahora en adelante. El recolector se dará cuenta de cuales no se usarán más y liberará la memoria asociada a ellos, de ahí su nombre.

Antes era el programador quien tenía que tener especial cuidado y hacer esta tarea pero ahora no tiene que preocuparse porque el recolector de basura es una aplicación que se activará cuando se quiera crear un objeto pero no se puede completar por falta de memoria. Entonces recorre la memoria dinámica asociada a esa aplicación detectando objetos inaccesibles (objetos basura) y los elimina.

7. Tratamiento de error en los tipos de datos. Los errores en los tipos de datos son un tipo de errores que a priori son simples pero que cuando se dan pueden llegar a ser difícil de detectar. Para evitar esto el CLR comprueba en todas las operaciones que los tipos origen y destino sean compatibles para todas las conversiones de tipos que se hagan en la ejecución.

8. Aislamiento de procesos. Otra gran característica del CLR es que aísla los procesos, es decir, que se asegura de que el código de un proceso no pueda acceder al código de otros evitando así errores de programación muy frecuentes como este tipo de ataques entre procesos involuntarios e imprevistos.

Esto se consigue gracias al sistema de seguridad de tipos antes comentado, pues evita que se pueda convertir el tipo de un objeto a uno de mayor tamaño que el suyo propio y que por tanto pueda por error acceder a espacios de memoria en principio ajenos a él.

Además el CLR tiene otra característica, y es que para acceder a la memoria tiene que tener una posición destino indicada, no se permite acceder a posiciones arbitrarias.

9. Tratamiento de errores, las excepciones. Al darse un error de ejecución en otros sistemas Windows, este se transmitía en forma de HRESULTS, excepciones u otros códigos en formato Win32. Sin embargo, ahora con el CLR y .NET, todos los errores que se den son notificados y se propagan de la misma manera, mediante excepciones.

Además y enlazándolo con otra gran característica de .NET de la que he hablado anteriormente como es la posibilidad de trabajo en .NET con diferentes lenguajes, el CLR permite tratar con un lenguaje excepciones que han sido lanzadas en un segundo lenguaje e incluso que a su vez han sido capturadas por otro tercer lenguaje diferente. Lo que dialécticamente puede liar al lector, si se entiende, facilita la tarea al programador que sabe trabajar con varios lenguajes y que utilizará cada uno de ellos para un propósito diferente.

10. Soporte multihilo. Dependiendo de los procesadores de la máquina que utilices, el CLR es capaz de trabajar con aplicaciones divididas en múltiples hilos de ejecución que se ejecutan por separado, en paralelo o intercalándose.

11. Distribución transparente. En otro intento de unir todos los sistemas, .NET ha apostado fuerte por los denominados “objetos remotos”, que no son más que objetos que se crean con el fin de ser compartidos y utilizados a distancia como si estuvieran situados físicamente en la propia máquina desde donde se trabaja.

El CLR ofrece la infraestructura necesaria para poder crearlos y acceder a ellos de manera absolutamente transparente.

12. Elección de código a ejecutar. El CLR proporciona mecanismos para restringir la ejecución de ciertos códigos o para no dar el mismo nivel de confianza a un código que proviene de internet como a uno

procedente de una red local, por ejemplo. De este modo se evita poner en peligro la integridad de los archivos al ejecutar un código malicioso.

13. Interoperabilidad con código antiguo. Una de las propiedades más importantes es que el CLR incorpora los mecanismos necesarios para poder acceder desde código nuevo escrito para la plataforma .NET a código escrito previamente y que por tanto no estaría preparado para ser ejecutada dentro de ésta como es el caso de los objetos COM y algunas funciones de DLL como la API Win32.

Microsoft Intermediate Language (MSIL o CIL)

Es un código intermedio en el que se basa toda la plataforma .NET y que es generado por el CLR cuando un compilador compila un código.

Una de las principales ventajas de .NET es que todos los compiladores que generan código para esta plataforma no lo hacen para unos tipos de CPU en concreto, si no que gracias a que lo hacen en MSIL sirve para todas. Por tanto, podemos decir que MSIL es un lenguaje de un nivel de abstracción mucho más alto que el de la mayoría de los códigos máquina de las CPUs existentes.

Sin embargo no todo son ventajas, las CPU no pueden ejecutar código MSIL directamente. Para hacer esto necesitan un compilador **JIT (just in time)** o más comúnmente llamado Jitter incluido en el CLR que convertirá este código MSIL al código de la nativo de la CPU donde se vaya a ejecutar.

Este es el esquema de compilación/ejecución de un código, los elementos y lenguajes que se generan:

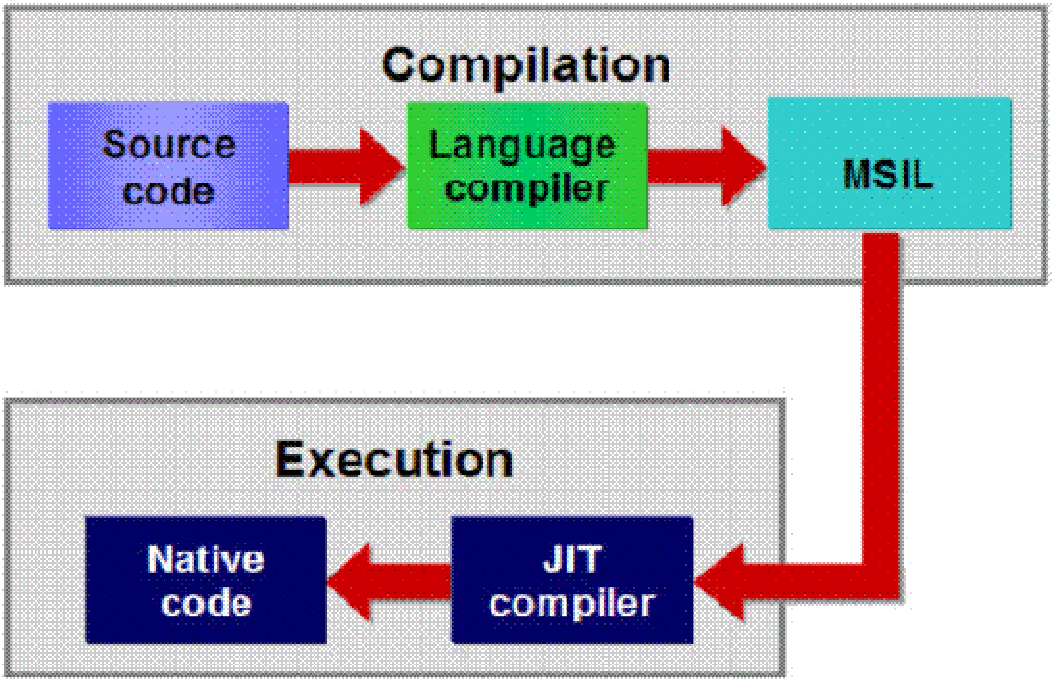


Ilustración 6. Esquema de pasos que sigue el código (C-Sharpcorner)

Como puede apreciarse en la imagen lo que ocurre es lo que hemos venido contando a lo largo de esta sección en la que hablo de .NET. El código fuente escrito en el lenguaje que usemos (C# en el caso de mi proyecto), pasa a ser compilado por el compilador, quien genera el lenguaje intermedio MSIL.

Una vez compilado el código se ejecuta en el CLR, pero como esto hemos dicho que está escrito en MSIL y no se puede hacer de manera directa, necesitaremos un compilador JIT o Jitter que convierta el código intermedio al código específico de la CPU donde se va a ejecutar hasta que finalmente tenemos el código máquina deseado para la arquitectura con la que estamos trabajando.

Este Jitter se distribuye en tres versiones:

- **Jitter normal:** Es el que se usa por defecto. Sólo compila el código MSIL a código máquina a medida que se necesite para ahorrar recursos. Así evita tener que compilar innecesariamente código que nunca se ejecute. En vez de llamar directamente a los métodos de las nuevas clases llama a funciones auxiliares denominadas "Stubs" que compilan el código real del método. Una vez hecho esto se sustituye el método correspondiente por una llamada directa a ese código compilado. Con esto lo que se evita es tener que llamar a todos los métodos de las nuevas clases, ahora lo que se hará es llamar directamente a ese único código del Stub que hemos compilado.

- **Pre Jitter:** Es una versión ya compilada que se distribuye como una aplicación en línea de comandos llamada "ngen.exe" mediante la que es posible compilar completamente cualquier ejecutable o librería que contenga código gestionado y convertirlo a código nativo. De este modo se ahorra tiempo en hacer una compilación dinámica.

- **Jitter económico:** El Jitter económico es la versión del Jitter normal para dispositivos empotrados con poca potencia de CPU y memoria porque es el más rápido. Es el Jitter que se usa habitualmente en sistemas como Windows CE. Funciona de forma similar al jitter normal solo que no realiza ninguna optimización de código al compilar, sino que traduce cada instrucción MSIL por su equivalente en el código máquina sobre la que se ejecute.

El Jitter hace que pueda parecer más lenta la ejecución de una aplicación gestionada debido al tiempo invertido en las compilaciones dinámicas. En este sentido plataformas como Java podrían ser más rápidas, sin embargo hay que tener en cuenta que .NET interpreta el código sólo una vez: cuando se llama al método al que pertenece, no cada vez que se ejecute, lo que a la larga resulta positivo.

Además como el CLR tiene al recolector de basura la memoria queda siempre compacta generando rapidez en la reserva de memoria (siempre y cuando haya).

Gracias a esto los ingenieros de Microsoft piensan que las venideras versiones de los Jitter podrían conseguir que el código gestionado sea ejecutado más rápido que el no gestionado, algo que por norma general no ocurre ahora mismo.

Resumen de funcionamiento general de .NET

Para aclarar finalmente cómo funciona .NET con su CLR, sus lenguajes específicos y el lenguaje intermedio MSIL he querido adjuntar esta imagen que creo que resume de manera perfecta cómo interactúan todos estos elementos y donde podemos situarlos físicamente.

Por eso la secuencia descrita a continuación debe compararse con la imagen para un mejor entendimiento.

- Primero escribimos el programa en el lenguaje que queramos usar (C#, VB.NET o J#, en el caso de la imagen).
- Lo compilamos por medio del propio compilador de este lenguaje que proporciona .NET. Generará un código intermedio llamado MSIL o CIL (Common intermédiate Language en inglés).
- Éste pasará a la parte que tienen en común todos los lenguajes. El CLR tratará este código y lo compilará a un código máquina que se podrá ejecutar en el ordenador en el que estás trabajando.

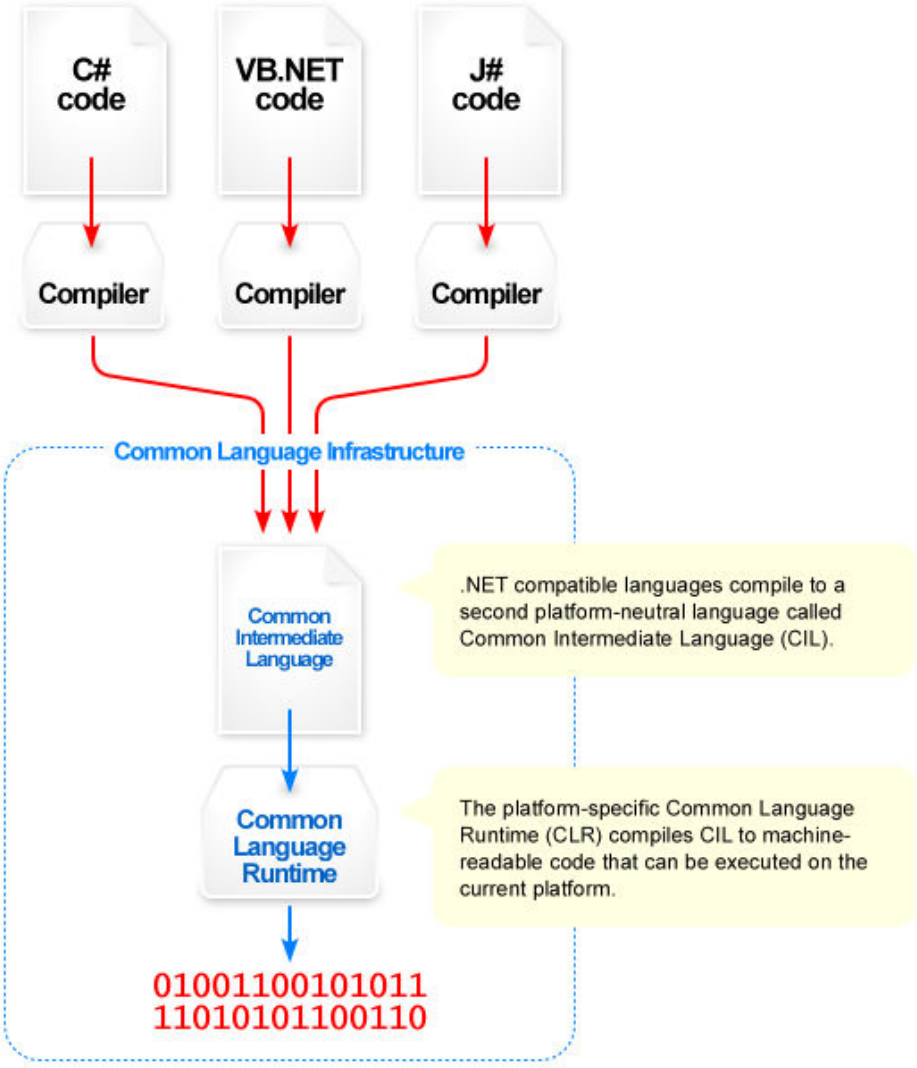


Ilustración 7. Esquema detallado de funcionamiento de .NET (codehaus.org)

Metadatos de .NET

Cuando se compila el código se hace en módulos o ficheros a los que se suele llamar **ejecutables portables** y que contienen la definición de los tipos de datos. Estos son:

- Los ejecutables (extensión **.exe**).
- Las librerías de enlace dinámico (extensión **.dll** generalmente).

La diferencia principal entre estos es que los ejecutables tienen un método que constituye un punto de entrada a partir del cual desde la línea de comandos del sistema operativo se puede empezar a ejecutar el código.

Cada uno de estos módulos contiene:

- **La cabecera de CLR**: Es un pequeño bloque de información que indica la versión de CLR que necesita, su firma digital, punto de entrada, etc.
- **Código MSIL**.
- **Los metadatos**. Definen los tipos de datos y sus miembros. Los metadatos de cada módulo los genera el compilador automáticamente al crearlo en forma de tablas. A modo de ejemplo voy a mostrar algunos en la siguiente tabla.

TABLA	Descripción
ModuleDef	Define las características generales del módulo. Almacena un identificador de versión de módulo que crea el compilador y el nombre que se dio al módulo al compilarlo.
TypeDef	Define las características de los tipos definidos en el módulo.
MethodDef	Define las características de los métodos definidos en el módulo.
ParamDef	Define las características de los parámetros definidos en el módulo.
FieldDef	Define las características de los campos definidos en el módulo.
PropertyDef	Define las características de las propiedades definidas en el módulo.
EventDef	Define las características de los eventos definidos en el módulo.
AssemblyRef	Indica cuáles son los ensamblados externos (ficheros) a los que se referencia en el módulo.

ModuleRef	Indica cuáles son los otros módulos del mismo ensamblado a los que referencia el módulo. De cada uno se indica cuál es su nombre de fichero.
TypeRef	Indica cuáles son los tipos externos a los que se referencia en el módulo. Referencia a la tabla AssemblyRef.

Tabla 2. Tablas de metadatos

Ensamblados en .NET

Hemos hablado durante todo el tema de la capacidad que tiene .NET de agrupación de elementos bajo un mismo techo, de modo que sólo tengamos que usar un tipo (que a su vez contendrá millones de ellos) y no esos mismos millones por separado. Pues bien, el ensamblado es una de estas técnicas que usa .NET y que hacen de él una plataforma sencilla y fácil de usar.

Un ensamblado es una agrupación lógica de uno o más módulos o ficheros de recursos (tales como archivos de imagen como .GIF, .JPG, páginas web .HTML, etc.) que se engloban bajo un nombre común con el objetivo de abstraernos y separar el uso de dichos ficheros de recursos con su ubicación física. De este modo tendremos un proyecto con numerosos elementos agrupados y localizados todos bajo un mismo nombre.

Por ejemplo, a la hora de optimizar recursos en internet podemos incluir todos los tipos de una aplicación en un mismo ensamblado. Dentro haremos una distinción: colocar los elementos más usados en un módulo y los menos usados en otro. De ese modo al navegar no tendremos que descargar los elementos más frecuentes porque ya están en un módulo de acceso específico y sólo tendremos que descargar aquellos que están en el módulo de los menos usados cuando queramos leerlos.

Todo ensamblado contiene un manifiesto, que contiene la información del ensamblado ordenado en tablas. Las principales incluidas en los manifiestos son las siguientes:

Tabla	**Descripción**
AssemblyDef	Define las características generales del ensamblado.
FileDef	Define cuáles son los archivos que forman el ensamblado.
ManifestResourceDef	Define las características de los recursos incluidos en el módulo.
ExportedTypesDef	Indica cuáles son los tipos definidos en el ensamblado y accesibles desde fuera del mismo.

AssemblyProccesorDef	Indica en qué procesadores se puede ejecutar el ensamblado. Suele estar vacío, lo que indica que puede usarse en cualquiera.
AssemblyOSDef	Indica bajo qué sistemas operativos se puede ejecutar. Suele estar vacía, lo que indica que se puede ejecutar en cualquier procesador.

Tabla 3. Tablas de manifiesto

Para asegurar que no se haya alterado la información de ningún ensamblado se usa el criptosistema de clave pública RSA, que es un sistema criptográfico de clave pública usada para firmar digitalmente.

Tipos de ensamblados:

- Ensamblados privados. Los privados se almacenan en el mismo directorio que la aplicación que los usa limitando su uso sólo a ésta.
- Ensamblados compartidos. Se almacenan en un caché de ensamblado global (GAC) y pueden usarlos cualquiera que haya sido compilada referenciándolos. Para mantener la seguridad han de ser cifrados con RSA.
  El GAC puede mantener múltiples versiones de un mismo ensamblado pero cuando una aplicación se ejecuta sólo puede usar una de las versiones del mismo, solucionando así un problema de la misma índole que el que comenté previamente con los archivos DLL.

Librería de clase base (BCL)

La Librería de clase base es un elemento fundamental de .NET que contiene los cientos de tipos de datos que podemos usar en .NET y que permiten acceder tanto a los servicios ofrecidos del CLR como a las funcionalidades más frecuentemente usadas a la hora de escribir programas

Hay que agregar que a partir de estas clases prefabricadas, el programador puede crear otras nuevas clases que mediante herencia extiendan su funcionalidad y se integren a la perfección con el resto de clases de la BCL.

Esta librería está escrita en MSIL, por lo que puede usarse desde cualquier lenguaje cuyo compilador genere MSIL.

Dado la amplitud de la BCL, ha sido necesario organizar los cientos de tipos posibles en los llamados “namespaces” que agrupen clases con funcionalidades similares. Los que más se usan y en concreto los que he particularmente he incluido en mi proyecto (en color verde) vienen definidos en la siguiente tabla:

NameSpace	Utilidad de los tipos de datos que contiene
System	Tipos muy frecuentemente usados, como los tipos básicos, tablas, excepciones, fechas, números aleatorios, recolector de basura, entrada/salida en consola, etc.
System.Collections	Colecciones de datos de uso común como pilas, colas, listas, diccionarios, etc.
System.Data	Manipulación de bases de datos.
System.IO	Manipulación de ficheros y otros flujos de datos.
System.Net	Realización de comunicaciones en red.
System.Text	Contiene las clases que representan codificaciones de caracteres de texto.
System.Runtime.Remoting	Acceso a objetos remotos.
System.Windows	Provee de todas las clases necesarias para trabajar con la tecnología WPF.
System.Data.SQLIte	Permite usar SQLite
Mnc.DetailedInfoPlugin	Contiene las clases específicas del sistema creado por MNC.
System.Winforms	Creación de interfaces de usuario basadas en ventanas para aplicaciones estándar.
System.XML	Acceso a datos en formato XML.
System.Windows.Shapes	Contiene las clases necesarias para representar formas que se usarán en el lenguaje XAML.
System.Windows.Media	Permite la integración de contenido multimedia rico tal como imágenes o vídeos en aplicaciones WPF.

Tabla 4. Espacios de nombres más usados en la BCL

Common Type System (CTS). Sistema de tipo común.

Al conocer los detalles que se han mostrado en apartados anteriores uno puede preguntarse cómo puede el CLR detectar si todos esos tipos de datos que existen son compatibles con él.

Precisamente para esto está el CTS, la parte de .NET que aporta esas reglas que tienen que cumplir los datos para que el CLR los reconozca como válidos.

Los CTS tienen modificadores de acceso que informan sobre qué se puede hacer esa clase y su visibilidad. Los admitidos son:

Modificador	Código desde el que es accesible el miembro
Public	Cualquier código.
Prívate	Código del mismo tipo de dato.
Family	Código del mismo tipo de dato o hijos de éste.
Assembly	Código del mismo ensamblado.
family and assembly	Código del mismo tipo o hijos de éste ubicados en el mismo ensamblado.
family or assembly	Código del mismo tipo o de hijos de éste, o código ubicado en el mismo ensamblado.

Tabla 5. Modificadores de acceso más comunes en el CTS

Common Language Specification (CLS)

A la hora de definir los tipos de datos han de seguir un conjunto de reglas. El **Common Language Specification** (CLS) o Especificación del Lenguaje Común es ese conjunto.

A continuación se listan algunas de reglas significativas del CLS.

- Los tipos de datos básicos admitidos son “bool”, “char”, “byte”, “short”, “int”, “long”, “float”, “double”, “string” y “object”.
- Las tablas han de tener un número de dimensiones fijas.
- Cada tabla ha de indexarse empezando a contar desde 0.
- Todas las excepciones han de derivar de System.Exception.
- Los métodos de acceso a propiedades en que se traduzcan las definiciones get/set de éstas han de llamarse de la forma **get_X** y **set_X** respectivamente, donde X es el nombre de la propiedad.

- En un mismo ámbito no se pueden definir varios identificadores cuyos nombres sólo difieran en las mayúsculas o minúsculas que tienen, de este modo se evitarán los problemas al trabajar con otros lenguajes que no sean sensibles a la capitalización.

3.2.2. WINDOWS PRESENTATION FOUNDATION (WPF).

INTRODUCCIÓN:

Tras una primera conversación con Miguel Ángel Zurdo, responsable de tecnología de MNC, decidí elaborar la interfaz de la aplicación de mi proyecto usando esta tecnología puesto que es más moderna y viene pisando fuerte en el campo del diseño de interfaces y aplicaciones. Además, según pude comprobar, está desbancando en cuanto a asignación de recursos por parte de Microsoft a la que para entonces era la tecnología más usada en el desarrollo de interfaces, los Windows Forms.

Miguel Ángel Zurdo me explicó que hacer una interfaz usando Windows Form sólo contribuiría a continuar con el tipo de interfaz que caracterizaba a los programas desarrollado por MNC, y que él creía conveniente mejorar ahora que existían tecnologías como WPF. Esa precisamente era una de sus metas, iniciar sus aplicaciones en esta tecnología que brinda muchas más posibilidades de cara a realizar aplicaciones con una interfaz más bonita y con más efectos.

A pesar de que en MNC nadie sabía usar WPF y que por tanto no podrían resolverme las dudas que me fueran surgiendo decidí aventurarme puesto que había leído interesantes cosas de esta nueva tecnología. Además como tecnólogo sé que hoy en día los usuarios tienen cada vez mayores expectativas en cuanto a la experiencia del uso de software y las compañías están gastando una gran cantidad de tiempo y dinero para que sus interfaces de usuario se diferencien de las de la competencia.

Se podría decir que es la era del interfaz y por eso Microsoft no ha querido quedarse atrás ante los competidores. Por eso ha inventado una solución para ayudar a las personas crear un software del siglo veintiuno que reúna estas altas exigencias utilizando menos tiempo y menos dinero.

WPF es una manera de crear interfaces con un innovador método: separar lo que es puramente la interfaz y el código que proporciona la funcionalidad a la aplicación, o lo que es lo mismo, la parte del diseño y del desarrollo.

Tanto es así que para crear la interfaz se utilizará un lenguaje de marcado como XAML y para la funcionalidad, el lenguaje .NET que uno prefiera; en mi caso C#.NET. De este modo un diseñador podrá realizar toda la interfaz de una manera fácil e intuitiva y que, sea esta la gran diferencia frente a sus competidores, podrá dar al desarrollador para que la lleve a cabo exactamente igual a como él diseñó. Porque antes existían grandes diferencias entre lo que el diseñador planeaba y lo que finalmente el desarrollador podía crear debido a las limitaciones tecnológicas. Pues bien, con WPF esto ha llegado a su fin puesto que ambos, diseñador y desarrollador, utilizarán el mismo archivo XAML y si el diseñador ha sido capaz de crearlo el desarrollador, asegura Microsoft, es capaz de plasmarlo en el programa.

WPF EN PROFUNDIDAD:

En 1973, los investigadores del Stanford Research Institute desarrollaron una interfaz de hipervínculos en modo texto gobernadas por un ratón, que dicho sea, también inventaron. Sin embargo fue Xerox la primera empresa en crear el primer ordenador personal con un denominado “escritorio” y una interfaz gráfica de usuario (GUI).

Esto no fue más que el principio. Más tarde IBM, Microsoft y Apple se lanzaron al mercado de los entornos basados en ventanas y han estado compitiendo (sobre todo estos dos últimos) en la consecución de las mejores interfaces junto a terceros, lo que ha hecho que se evolucione en apenas 30 años de manera sobrenatural en la experiencia que estas ofrecen.

A principios de 1990, OpenGL se convirtió en una biblioteca de gráficos populares para hacer 2D y gráficos avanzados 3D en Windows y otros sistemas diferentes. Esto fue aprovechado por la gente que crea diseño asistido por ordenador (CAD), programas de visualización científica, y juegos.

Más tarde apareció DirectX, una tecnología de Microsoft introducida en 1995 que proporcionaba una alternativa de alto rendimiento para gráficos en 2D, entrada, comunicación, sonido e incluso gráficos 3D (posible gracias a DirectX 2 en 1996). A partir de entonces ha ido mejorando hasta haber desarrollado su última versión, la denominada DirectX 11.

Con la introducción de. NET y el código administrado en el año 2002, los desarrolladores fueron agasajados con un modelo altamente productivo para crear aplicaciones para Windows y Web. En este mundo, Windows Forms se convirtió en la principal forma para C#, Visual Basic, y (en menor grado) C++, de crear nuevas interfaces de usuario en Windows. Pero todavía tiene todas las limitaciones fundamentales de GDI + y de USER.

A raíz de DirectX 9 Microsoft creó un marco para código administrado, el marco XNA. A pesar de que esto permitía a los programadores de C# usar DirectX sin la mayor parte de las complicaciones de .NET/COM, estos marcos administrados no eran tan fácil de usar a no ser que estuvieras programando un juego (entonces sería más fácil porque dicho marco incluía nuevas librerías específicas para desarrollo de juegos). (interfacesymposia.org)

Así que como se puede comprobar, en los años 90 era muy difícil crear aplicaciones informáticas con animaciones avanzadas para esos años. Un ejemplo es la interfaz de usuario que ofrecía Windows 3.0 allá por aquellos años de la imagen que muestro a continuación. Ruego presten atención a la interfaz del mundialmente conocido juego llamado “Reversi”.

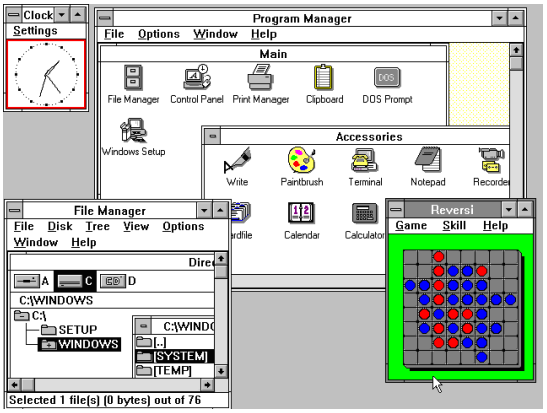


Ilustración 8. Interfaz de Windows 3.0 (Finplanet.ru)

Hay varias razones para explicar este déficit a lo largo de los años:

- El Hardware requerido para obtener esa buena experiencia de usuario no había sido creada hasta ahora. La competencia durante años ha hecho que evolucione exponencialmente.
- El Hardware dedicado a los gráficos era cada vez mejor y más barato, pero de poco servía mientras las herramientas para aprovecharlo no cambiaran. Gracias a eso la evolución fue mucho más lenta.

Hay que destacar que aparte de las mejoras tecnológicas hay un incremento en el interés por parte de desarrolladores y diseñadores en crear alternativas para las interfaces que se nos dan con las tecnologías más comunes. Hoy en día tratamos de hacer nuestros propios botones o marcos para diferenciarnos del resto en nuestras aplicaciones. Sin embargo, esto puede ser costoso y presentar problemas como que las aplicaciones sean tan accesibles, que no puedan ejecutarse si no se tienen determinados privilegios, que no rendericen bien en escritorios remotos o que funcionen lentamente o produciendo errores visuales, entre otros.

Por eso Microsoft reconoció que se necesitaba algo nuevo que superara en fiabilidad y universalidad a GDI+ y USER; así nació WPF, la respuesta para los desarrolladores de software y diseñadores gráficos que deseen crear experiencias de usuario modernas sin tener que dominar tecnologías difíciles. (Nathan, 2006).

Principales características de WPF

1. Integración de elementos multimedia:

Antes de WPF un desarrollador de Windows que quería utilizar 3D, video, voz, visualizar documentos, gráficos en 2D y controles tendría que aprender varias tecnologías independientes con una serie de inconsistencias entre ellas e intentar mezclarlas sin soporte asociado (o con muy poco).
Con la mejora de .NET y su afán de agrupar todo bajo un mismo techo, WPF entra en el mercado cubriendo todas estas necesidades creando un modelo de programación coherente así como un ajuste integral de cada tipo de material. Además muchas de las técnicas que se aprenden en un área se aplican a otras muchas áreas.

2. Independencia en la resolución:

Con WPF se consigue que la calidad de las texturas no se modifique al ampliar o alejar la vista de las ventanas donde se ejecutan las aplicaciones.

Con WPF da igual crear aplicaciones que sean vistas tanto en un teléfono móvil como en una pantalla LCD de 60 pulgadas gracias al énfasis que han puesto en los gráficos vectoriales. El éxito en la mejora de esta tecnología puede ser fácilmente visto con la aplicación de “ampliación de pantalla de Windows” que incluye la adaptación con gráficos vectoriales.

La figura que muestro a continuación es un ejemplo de la aplicación de Windows “ampliación de pantalla”.

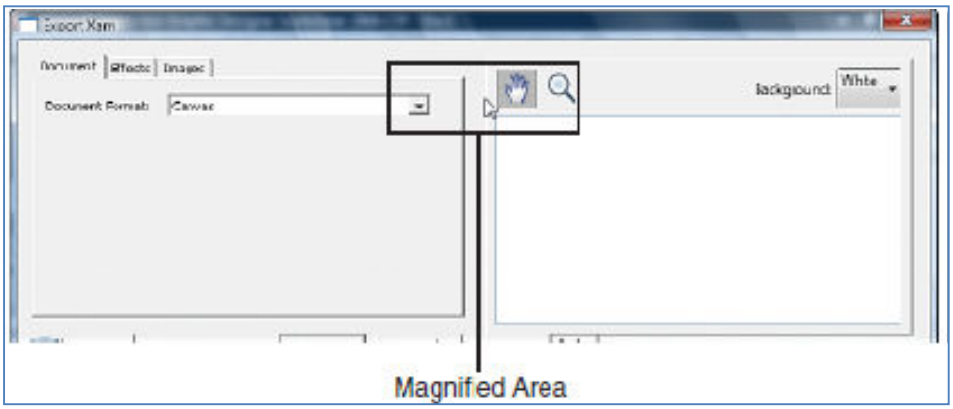


Ilustración 9. Ampliación de pantalla

En la siguiente imagen se muestra la ampliación de la zona delimitada por el rectángulo negro que se puede observar en la imagen superior. Como se puede apreciar, la parte izquierda pierde calidad en las líneas debido a esta ampliación y a que no está implementada con WPF mientras que la de la derecha sí lo está. Uno puede notar la diferencia de suavidad en el escalado de los botones de ambas partes mirando las líneas o la misma flecha del menú desplegable.

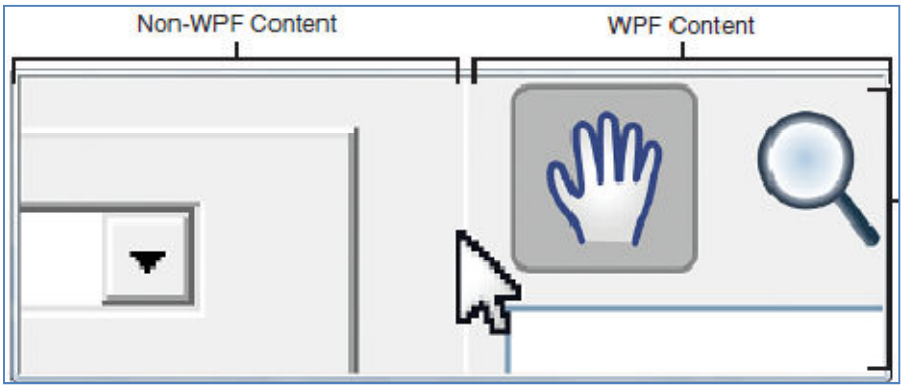


Ilustración 10. Diferencia entre contenido sin implentar e implementado en WPF

3. Aprovechamiento del hardware.

Cualquier contenido de WPF, ya sean gráficos o textos en 2D o 3D es convertido en triángulos 3D, texturas y otros objetos de Direct3D que luego serán renderizados mediante hardware, consiguiendo la mejora.

Por tanto, al contrario que sus competidores los sistemas basados en GDI, las aplicaciones creadas con WPF gozan de los beneficios de la aceleración mediante hardware para tener gráficos más suaves y un mejor comportamiento del ordenador puesto que los recursos que consume serán GPUs (Graphic processing unit) en vez de CPUs (central processor units), que entonces pueden dedicarse a otras funciones.

Esto además asegura que las aplicaciones WPF y no sólo los videojuegos reciben los beneficios asociados de los avances tecnológicos en hardware y en drivers. Sin embargo no hay que engañarse, WPF no requiere un hardware de última tecnología porque es realmente su propio software de renderizado quien realiza las operaciones más importantes, posibilitando características no soportadas antes por hardware como la alta fidelidad a la hora de imprimir cualquier cosa por pantalla.

4. Uso de la programación declarativa.

Durante más de veinte años los programas basados en Win16 y Win32 han usado scripts declarativos para definir cuadros de diálogo y menús, aspecto que ha cambiado en la actualidad.

A pesar de que Windows Forms no ofrecía por defecto soporte para interfaces declarativas definidas por el usuario, los programas .NET de todos los tipos sí ofrecen una configuración y fuentes de un nivel superior basadas en lenguajes XML.

Sin embargo WPF va más allá del XML, lleva la programación declarativa a un siguiente nivel con la inclusión del XAML (Extensible Application Markup Language, pronunciado como “Zammel”), que usa para representar objetos. Por tanto la combinación de WPF y XAML para definir una interfaz de usuario es similar a usar HTML pero con un rango increíblemente más grande de expresividad.

El resultado es una menor distancia entre los diseñadores gráficos y quienes finalmente programan estos diseños.

5. Facilidad de aprendizaje:

Los controles en WPF son extremadamente fáciles de usar, no hay necesidad de formarse en profundidad en un lenguaje, ahora con XAML y los generadores de código es más fácil crear una interfaz.

Por estas 5 estas razones y otras muchas más se está consiguiendo que WPF sea hoy por hoy un referente en el diseño de interfaces porque combina los mejores atributos de sistemas como DirectX (3D y la aceleración de hardware), Windows Forms (muy productivo y eficaz al desarrollar), Adobe Flash (muy potente para animaciones) y HTML (marcado declarativo y fácil implementación).

Diferencias entre WPF y DirectX.

DirectX es más apropiado que WPF para desarrolladores avanzados escribiendo código para juegos o aplicaciones con modelos 3D complejos donde se necesita un gran rendimiento porque ofrece una interfaz de bajo nivel a los gráficos del hardware y puede ser complicado. Esto es pesado, si, pero a su vez permite que los programadores expertos realicen excelentes trabajos.

En contraste con DirectX, WPF proporciona un alto nivel de abstracción que hace que puedas tener una descripción de la escena y sugiere la mejor forma de renderizarlo una vez dados los recursos disponibles en tu computadora.

Otra gran diferencia es el coste de desarrollo, con WPF disminuye en gran cantidad debido a que se tienen que hacer muchas menos pruebas con los drivers con los que se va a trabajar porque Microsoft

ha invertido mucho tiempo en hacer ya esas pruebas de hardware antes. De este modo uno puede invertir su tiempo y dinero en mejorar otros apartados de tu aplicación.

Otra ventaja de WPF sobre DirectX es la mejora que se da sobre las aplicaciones a través de escritorios remotos, aprovechando mejor los GPUs.

Por último y como contrapunto a esta "competición" entre estas dos tecnologías podemos decir que a la hora de crear una aplicación se pueden usar ambas.

WPF VS Windows Forms. ¿Cuál usar?

A día de hoy Windows Forms no está muerto pero es verdad que se puede decir que Microsoft "lo ha abandonado" asignando una mayor cantidad de recursos a WPF en detrimento de este. Así pues se puede considerar que WPF es la evolución de Windows Forms.

Sin embargo, en el mundo empresarial donde no se tiene mucho cuidado con las interfaces y donde se busca muchas más veces funcionalidad y rapidez antes que una bonita interfaz, muchas personas opinan que Windows Forms es la mejor elección para crear aplicaciones de empresa tradicionales. Sin embargo los expertos coinciden en que esta gente se equivoca puesto que con WPF no sólo podemos hacer interfaces novedosas, además podemos hacer interfaces tradicionales exactamente igual que las que harías con Windows Forms.

Por estas razones y otras muchas y a no ser que estés usando Windows 98 o anteriores (que no soportan WPF) recomiendo siempre usarlo antes que Windows Forms. (JohnSmithOnWPF)

WPF VS Adobe Flash

En la creación de contenido web, Flash es actualmente la opción más popular debido a su capacidad de ubicación, es decir, que puedes poner contenido flash en una página con la seguridad de que la mayoría de los visitantes van a tener instalado el reproductor necesario, y en caso de que no lo tengan, es muy fácil descargarlo.

Pero Flash no es el único en este mercado, WPF también puede ejecutarse en las webs y tiene además la ventaja de tener mejores herramientas y modelo de programación, un robusto control de reutilización y mejor soporte del lenguaje de programación.

En contra hay que objetar que para poder ver dicho contenido de WPF en la web se requiere Windows y la plataforma de trabajo .NET framework 3.0 como mínimo que, aunque viene instalado por defecto en Windows Vista y posteriores, no viene instalado o no existe actualmente para otros sistemas, con lo cual se limita mucho el número de usuarios que pueden disfrutar de ello.

Precisamente para combatir este problema Microsoft ha presentado WPF/E (Windows Presentation Foundation Everywhere), una versión de WPF pero de pequeño tamaño y fácil de instalar en todos los sistemas. Soporta XAML y JavaScript además de C# y Visual Basic. Se espera que WPF/E soporte gráficos basados en vectores, imágenes, video, animación, texto y controles básicos, pero no soportará 3D, soporte de documentos rico, extensibilidad o aceleración hardware. WPF/E nace como una alternativa a Flash.

WPF como parte del framework .NET

WPF es un componente importante del Framework o marco de trabajo .NET. A la mayoría de los usuarios les choca que la primera versión de WPF sea la 3.0 y no la 1.0. Esto es debido a la absoluta dependencia de WPF con .NET ya que fue en esta versión de .NET donde apareció la primera de WPF.

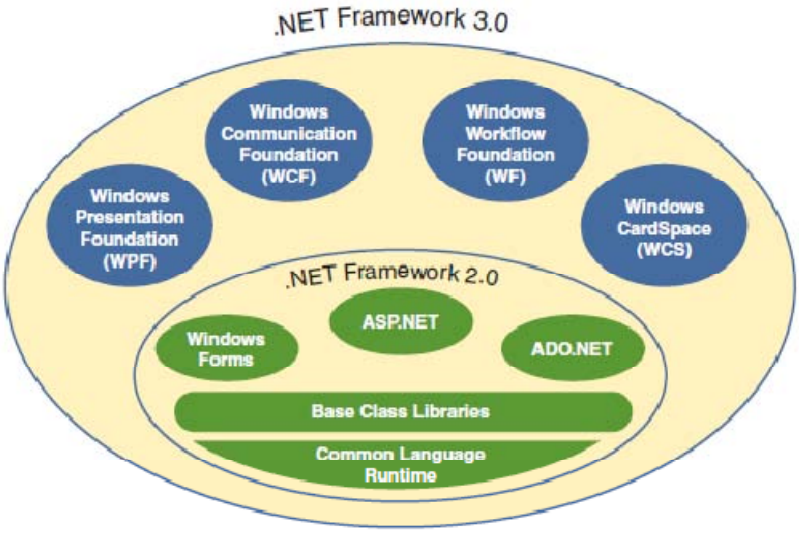


Ilustración 11. Esquema de evolución de .NET 2.0 a su versión 3.0

El Framework .NET 3.0 es claramente una evolución del Framework .NET 2.0, su precursor, como puede apreciarse en la imagen anterior. Las tecnologías en el pequeño círculo inferior no han cambiado, lo que ha ocurrido es que para la versión 3.0 se han añadido 4 nuevas: WPF (Windows Presentation Foundation), WCF (Windows Communication Foundation), WF (Windows Workflow Foundation) y WCS (Windows CardSpace).

Siguiendo la filosofía de .NET, todas esas tecnologías son compatibles con cualquier lenguaje de programación .NET (denominados normalmente Managed Languages o lenguajes administrados) como C#, Visual Basic o C++/CLI entre otros muchos.

Enfatizando en las mejoras que esta evolución supone cabe destacar el uso de XAML y XML para expresar funcionalidad en el diseño de las interfaces de usuario. Ambos lenguajes están llamados a ser ese lenguaje común que se supone que todas las partes hablarán algún día, sobre todo en la comunicación entre diseñadores y desarrolladores.

Pero XAML no sólo es usado en WPF, sino que por ejemplo también es usado en WF (Windows Workflow Foundation) para representar diagramas de flujo o en WCF (Windows Communication Foundation) para comunicarse con otros programas.

Nuevos e importantes conceptos que introduce WPF

En esta sección quiero mostrar al lector algunos de los nuevos conceptos que WPF introduce al marco de trabajo .NET.

1. Árboles lógicos y visuales.

WPF representa con XAML interfaces de usuario de un modo natural debido a su naturaleza jerárquica. En WPF las interfaces de usuario están construidas a partir de un árbol de objetos denominados "Árboles lógicos". El código mostrado a continuación muestra un ejemplo de estos árboles lógicos donde la raíz es una ventana.

```
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Title="NewConcepts"
SizeToContent="WidthAndHeight"
Background="Green">
        <StackPanel>
                <Label FontWeight="Bold" FontSize="20" Foreground="White">
                        WPF  3.0
                </Label>
                <Label>© Vicente Garcia</Label>
                <Label>Capítulos disponibles:</Label>
                <ListBox>
                        <ListBoxItem>Capitulo 1</ListBoxItem>
                        <ListBoxItem>Capitulo 2</ListBoxItem>
                </ListBox>
                <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
                        <Button MinWidth="75" Margin="10">Ayuda</Button>
                        <Button MinWidth="75" Margin="10">OK</Button>
                </StackPanel>
                <StatusBar>Has registrado este producto.</StatusBar>
        </StackPanel>
</Window>
```

Fragmento de código 4. Árbol lógico de una ventana en XAML

La ventana contiene un elemento "StackPanel" que es su hijo y que a su vez contiene unos cuantos controles simples como etiquetas de tipo String (WPF 3.0, Vicente Garcia y Capítulos disponibles), una "Listbox", otro "StackPanel" con botones y por último una "Status Bar".

Este es el resultado de ejecutarlo con xamlPad (herramienta instalada con Microsoft Visual Studio 2008) y su árbol lógico asociado es el que muestro a continuación.

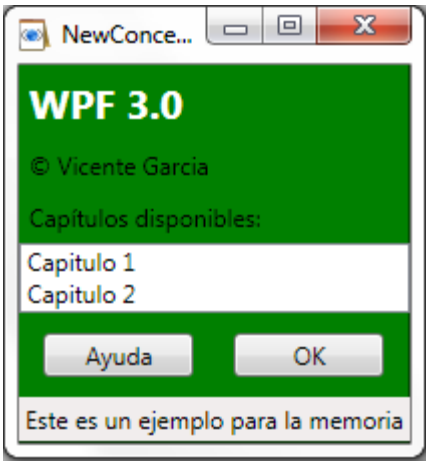


Ilustración 12. Resultado de un árbol lógico de una ventana

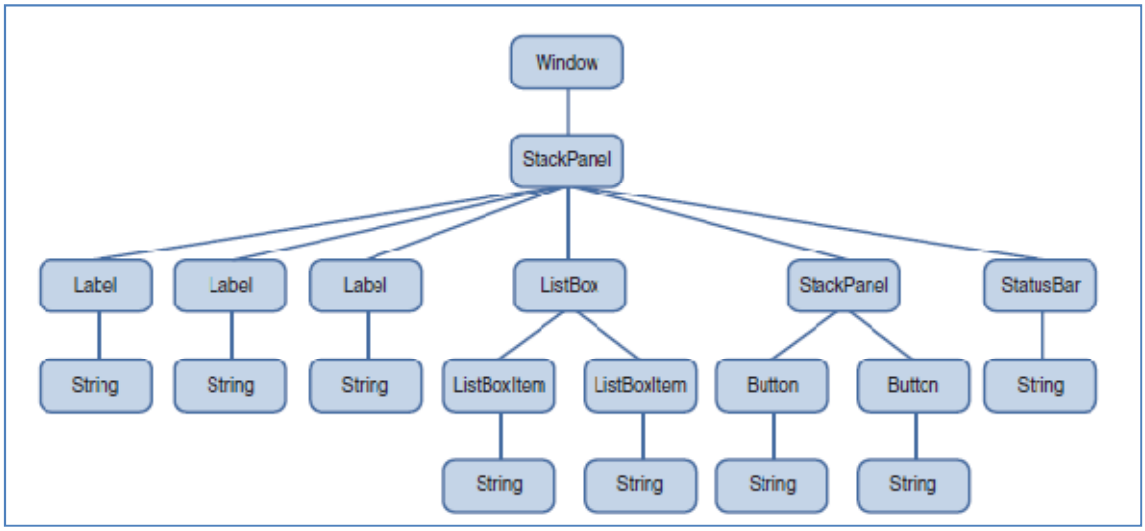


Ilustración 13. Árbol asociado a la ventana

El concepto de estos árboles es muy importante porque engloba cualquier aspecto de WPF (propiedades, eventos, fuentes) y dota al desarrollador de un enfoque gráfico y esquemático para agrupar los elementos del proyecto. Estos árboles pueden ser muy útiles por ejemplo para seguir los valores de las propiedades que por herencia son propagados a los niveles inferiores y también a los eventos pueden ir hacia los niveles superiores.

Un concepto muy similar al árbol lógico es el "árbol visual", que es básicamente una expansión del árbol lógico cuyos nodos se descomponen más detalladamente y reflejan sus componentes de menor nivel. Un árbol visual por tanto expone una visión más detallada de cara a la implementación.

Por ejemplo, a pesar de que un elemento "ListBox" es considerado un control simple en un árbol lógico, su representación en el árbol visual quedaría compuesta de más elementos de WPF tales como el borde, las ScrollBars y algunos otros.

Aquí os muestro el ejemplo de cómo quedaría representado el árbol visual de una etiqueta cualquiera, elemento todavía más simple que un "ListBox".

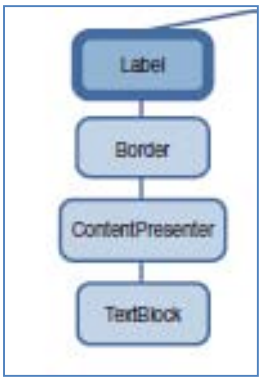


Ilustración 14. Árbol visual de una etiqueta

Hay que reseñar que sólo los elementos que derivan de de las librerías System.Windows.Media.Visual o System.Windows.Media.Visual3D son representables en un árbol visual, de modo que el contenido de un String, por ejemplo, no se incluye puesto que no tiene un comportamiento jerárquico por sí mismo.

Los árboles visuales debido a todas estas características son muy complejos y aunque sirven de gran ayuda muchas veces sólo creo que es necesario hacerlos en dibujos donde se requiera una precisión y un seguimiento milimétrico que sólo se da en proyectos donde se trabaja a muy bajo nivel.

WPF y las posibilidades de formato

La clave de que WPF sea tan declarativa y variable es el uso de las propiedades. Por ejemplo, un botón puede tener hasta 96 diferentes que modificaremos a antojo del diseñador para darle color, forma, transparencia, tamaño, etc.

Estas propiedades pueden ser fácilmente configuradas en XAML directamente o con alguna herramienta de diseño sin ningún tipo de código basado en funciones. En mi caso he usado Microsoft Expression Blend 3.0 aunque podría haber utilizado Microsoft Visual Studio 2008.

Conclusión

Cuanto más tiempo pasa más software de calidad aparece, algunos con apariencia casi cinemática y otros que no toman riesgos y parecen pasados de moda. De cualquier modo el esfuerzo que supone crear interfaces de usuario (especialmente en Windows) es ahora mucho más sencillo que en el pasado gracias a WPF, donde es mucho más fácil crear un programa que trabaja con experiencias en tres dimensiones y contenido multimedia como programas que tienen un estilo más "Retro".

Con WPF también los diseñadores gráficos son expertos. Ahora pueden usar herramientas de diseño que proporcionan una gran expresividad sin tener que tener en cuenta el lenguaje en el que se va a

programar. Al usar los diseñadores gráficos esta tecnología se consigue que se puedan entender a la perfección con los programadores de código, haciendo que no haya diferencia entre lo que unos diseñan y los que finalmente otros desarrollan; problema que por otro lado se ha dado siempre.

Con estos capítulos he explicado brevemente como funciona WPF y con ello la parte del marco de trabajo .NET que voy a usar.

Y es que Microsoft podría haber desarrollado WPF como cualquier otro lenguaje formal pero en vez de esto ha apostado por una nueva técnica que puede aportar una gran productividad a diseñadores y desarrolladores en común.

3.2.3 C.NET# (C_SHARP)

INTRODUCCIÓN

El entorno de trabajo en esta empresa es .NET pero el lenguaje de programación con el que se trabaja es C#.NET, de ahí que el requisito principal de este proyecto sea desarrollar mi aplicación en dicho lenguaje orientado a objetos de propósito general.

En .NET se podría trabajar con prácticamente cualquier lenguaje pero a pesar de esto Microsoft decidió crear con C# un nuevo lenguaje que fuera nacido para ser usado en este marco de trabajo y así poder aprovechar toda su potencia, cosa que no se llegaba a conseguir con las derivaciones de los otros lenguajes usadas en .NET.

El creador de C# es el mismo de otros muchos entornos y lenguajes como Turbo Pascal, Visual J++ o Delphi. Por eso C# combina las mejores características de unos y otros que hacen de él un lenguaje sencillo y completo.

Para aprender a manejar C# hace falta formarse aunque es más fácil de aprender si previamente se han visto lenguajes como C++, Java u otros orientados a objetos para poder hacerse rápido con estos nuevos conceptos diferentes a la programación estructurada. (Desarrolloweb.C#)

C# EN PROFUNDIDAD

C# (leído en inglés C Sharp) es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Se puede considerar el lenguaje nativo de .NET puesto que carece de elementos innecesarios heredados en .NET y es más sencillo e intuitivo que el resto de los lenguajes que se pueden usar.

La sintaxis y estructuración de C# es muy similar a la C++ ya que una de las principales intenciones de Microsoft era el facilitar la migración de códigos escritos en los otros lenguajes a C#.

Hay que destacar que en un principio Microsoft intentó que el lenguaje nativo de .NET fuera Java pero hubo usa serie de problemas con Sun Microsystems, empresa desarrolladora de este, y por tanto Microsoft se vio obligado a desarrollar su propio lenguaje donde mejoraría lo visto creando un lenguaje orientado al desarrollo de componentes.

En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Java, Visual Basic o C++ y las combina en uno solo. (Jett Fergusson, 2003)

Características principales de C#

Con la idea de que los programadores más experimentados puedan obtener una visión general del lenguaje recojo a continuación de manera resumida las principales características de C#. Algunas no es que sean sólo aplicables a C# sino que lo hacen a la plataforma .NET puesto que tienen repercusión directa sobre este.

1. C# es un lenguaje moderno que conserva virtudes de lenguajes antiguos

C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular (caso de la inclusión de los tipos decimal, string o bool o de instrucciones como foreach para recorrer colecciones).

2. Sencillez

C# elimina muchos elementos y propiedades que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo, el código escrito en C# es autocontenido, lo que significa que no necesita ficheros adicionales al propio fuente tales como ficheros de cabecera.

3. Está orientado a objetos

Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores **public**, **private** y **protected**, C# añade un cuarto modificador llamado **internal**, que puede combinarse con **protected** e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

Respecto a la herencia, a diferencia de C++ y al igual que Java, C# sólo admite la simple de clases.

4. Facilidad en el trabajo con los objetos

La sintaxis de C# permite definir cómodamente **propiedades** (campos de acceso), **eventos** (asociación controlada de funciones de respuesta a notificaciones) o **atributos** (información sobre un tipo o sus miembros).

5. Seguridad en los tipos de datos

Tal y como he comentado en el apartado de .NET, C# incluye estos mecanismos que permiten asegurar que los accesos y las conversiones de los tipos de datos siempre se realicen correctamente, lo que permite evitar que se produzcan errores difíciles de detectar por acceso a memoria.

Por ello además de sólo admitir conversiones entre tipos compatibles como he comentado antes, no permite el uso de variables no inicializadas (el propio compilador da a los campos un valor por defecto), comprueba el tamaño de una tabla para poder acceder siempre al contenido dentro de esta, controla la producción de desbordamientos en operaciones aritméticas con la creación de excepciones, incluye unos punteros orientados a objetos llamados delegados que almacenan referencias a varios métodos a la vez y comprueban los tipos de retorno de estos.

6. Instrucciones seguras

En C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes como por ejemplo que las estructuras “Switch” tengan que acabar con una sentencia “break” o “goto” donde se indique la siguiente sentencia a ejecutar.

7. Sistema de tipos unificado

A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada **System.Object**, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos"). Esto facilita el diseño de colecciones genéricas para almacenar objetos de cualquier tipo.

8. Compatibilidad

Para facilitar la migración de programadores, el tamaño de los tipos de datos básicos es fijo y C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32.

3.2.4. XML.

INTRODUCCIÓN

XML es una tecnología sencilla que basa su aplicación en otras que la complementan. Es una manera de representar la realidad con la peculiaridad de que permite compartir los datos con los elementos con los que trabaja a todos los niveles y por todas las aplicaciones. Es universal y por tanto es muy importante hoy por hoy puesto que se tiende a una globalización de información y sistemas, cosa que ya se empieza a conseguir con el framework .NET.

XML además permite ahorrar tiempo al programador puesto que la validación de estos archivos ya lo hace el lenguaje en el que se está trabajando.

Es muy interesante en el mundo de Internet y el e-business, ya que existen muchos sistemas distintos que tienen que comunicarse entre sí y con XML es fácil hacerlo.

Para este proyecto utilizo el XML a la hora de crear esos paquetes de datos (archivos XML) con la información del ordenador donde está instalado el software. Información como el número de procesadores, su velocidad, porcentaje de disco libre o estado de la red entre otros, son reflejados en estos archivos XML de los que extraeremos la información con la que rellenaremos la base de datos. Desde ella, posteriormente se leerán los datos para ser mostrados desde la aplicación que he creado para pantalla táctil.

XML EN PROFUNDIDAD

El lenguaje XML proviene de un lenguaje que inventó IBM en los años setenta. Este lenguaje se llamaba al principio GML (General Markup Language) y surgió por la necesidad que tenían en esta gran empresa de almacenar grandes cantidades de información de temas diversos. De este modo los expertos se inventaron GML, un lenguaje con el que poder clasificarlo todo y escribir cualquier documento para que se pueda luego procesar adecuadamente.

Este lenguaje gustó mucho al grupo ISO, una entidad que se encarga de normalizar todos los procesos de usuario, de modo que, cerca del año 1986 trabajaron para normalizar el lenguaje creando el SGML, que no era más que el GML pero estándar. SGML era un lenguaje muy trabajado, capaz de adaptarse a un gran abanico de problemas y a partir de él se han creado los siguientes sistemas para almacenar información.

En el año 1998 el grupo W3C, empezó el desarrollo de XML (Extended Markup Language). Hoy en día, muchas personas con grandes conocimientos en la materia están trabajando todavía en la gestación de XML. Con esto se pretende solucionar las carencias de HTML en lo que al tratamiento de información respecta como pueden ser que HTML no permita compartir información con todos los dispositivos como teléfonos móviles, que la presentación en pantalla depende del visor que se utilice.

XML es una tecnología en realidad muy sencilla que tiene a su alrededor otras tecnologías que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. De este modo XML representa una manera distinta de hacer las cosas más avanzada cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles y por todas las aplicaciones y soportes.

Así pues, XML juega un papel importantísimo en este mundo actual que tiende a la globalización y la compatibilidad entre los sistemas ya que es la tecnología que permitirá compartir la información de manera segura, fiable y fácil. Además, XML permite al programador dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de estos o el recorrido de las estructuras corren a cargo del lenguaje y está especificado por el estándar, liberando al programador de esta tarea.

XML es interesante en todo el mundo de la informática pero actualmente y sobre todo en el mundo de Internet y el e-business ya que existen muchos sistemas distintos que tienen que comunicarse entre sí usando este lenguaje. (Desarrolloweb.XML)

XML se escribe en un documento de texto ASCII, igual que el HTML y en la cabecera del documento se tiene que poner el texto

```
<?xml version="1.0"?>
```

Fragmento de código 5. Cabecera de un archivo XML

En el resto del documento se deben escribir etiquetas de forma anidada como las de HTML, donde se permite incluir tantos atributos como el usuario necesite.

Las etiquetas tendrán la forma:

```
<ETIQ1>...
        <ETIQ2>...
        </ETIQ2>...
</ETIQ1>
```

Un ejemplo de etiqueta sería:

```
<ETIQ edad="26" nombre="Padre Oso">
```

Fragmento de código 6. Etiqueta en XML

A modo informativo voy a crear una ficha XML que pueda representar una película con algunas de sus características más importantes.

```
<?xml version="1.0"?>
<PELICULA nombre="El show de Truman" año=1998>
        <PERSONAL>
                </DIRECTOR nombre="Peter Weir">
                </INTERPRETE nombre="Jim Carrey" interpreta-a="Truman Burbank">
                </INTERPRETE nombre="Laura Linney" interpreta-a="Meryl">
        </PERSONAL>
        </ARGUMENTO descripción="El reality show creado con la vida de una persona">
</PELICULA>
```

Fragmento de código 7. Ejemplo de una película en XML

3.2.5. XAML

INTRODUCCIÓN

En este proyecto trabajo principalmente con XAML, que es un lenguaje derivado del XML y que ha sido creado por Microsoft para trabajar sobre todo con la plataforma WPF dentro del marco de trabajo .NET. Los archivos XAML, vistos como texto, son en realidad archivos XML de marcado con ligeras diferencias y que se suelen codificar con sistema UTF-8.

XAML es un lenguaje declarativo de marcado como XML que simplifica la creación de la interfaz de usuario para una aplicación de .NET. Representa la creación de instancias de objetos y aporta un flujo de trabajo donde la aplicación y la interfaz pueden trabajar a la par.

En el caso de WPF creamos los elementos visibles de la interfaz con XAML declarativo y luego utilizamos un código subyacente, en el caso de este proyecto C# para crear la funcionalidad de la aplicación.

En este sistema, el código XAML ha sido generado con Microsoft Expression Blend, si bien ha sido modificado manualmente en muchas ocasiones tanto para reordenar parámetros como para completarlo con atributos que no incluía el programa. (XAML)

XAML EN PROFUNDIDAD

XAML, tal y cómo he hablado anteriormente es una importante herramienta para integrar a los diseñadores en el proceso de desarrollo del programa. Incluso si uno no tiene pensado trabajar con diseñadores debería aprender o por lo menos que le sea familiar este lenguaje por las siguientes razones:

- XAML es actualmente la forma más consistente para representar interfaces de usuario y otros objetos con sus jerarquías en .NET.
- El uso de XAML plantea una separación entre la apariencia del programa y lo que hay detrás, lo que ayuda mucho a la hora de realizar el mantenimiento del programa.
- El código XAML puede ser pegado en herramientas como XamlPad (en Windows SDK) para ver los resultados sin tener que compilar el código.
- XAML es el lenguaje en el que están escritas casi todas las herramientas de WPF, de aquí la importancia en este proyecto. (Nathan, 2006)

Antes de este tipo de lenguajes la comunicación entre diseñador y desarrollador podía seguir este esquema:

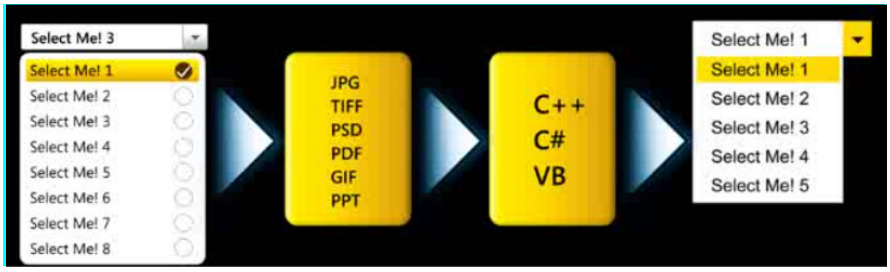


Ilustración 15. Esquema antiguo diseñador-desarrollador

Donde se puede apreciar que el formato de la pestaña desplegable que programa el desarrollador no es exactamente el mismo que el que el diseñador ha creado en un principio además de que no utilizará los mismos formatos de archivos y objetos para crearlo.

Sin embargo cuando surgió XAML ese mismo esquema podríamos decir que cambió a este otro:

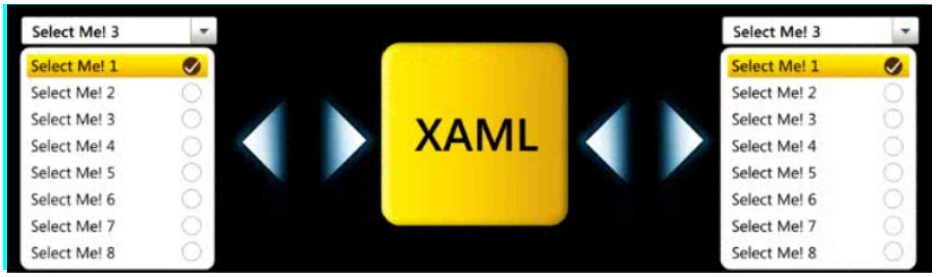


Ilustración 16. Esquema actual diseñador-desarrollador

Donde apreciamos que usando XAML y sus propiedades, atributos y objetos en común el resultado de la pestaña desplegable es exactamente el mismo para diseñadores como para desarrolladores.

Para crear una aplicación XAML es conveniente seguir los pasos del proceso que muestro a continuación:

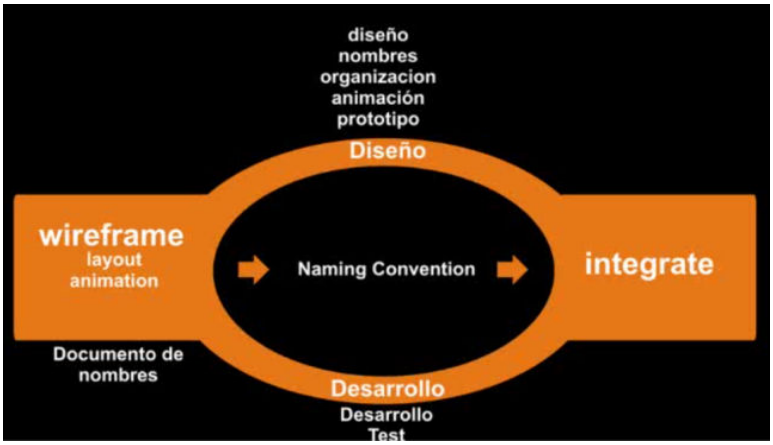


Ilustración 17. Proceso para crear una aplicación en XAML

- Crear el Wireframe: El primer paso. Es donde hacemos un diseño de lo que el cliente quiere. Para ello se crea un documento especificando el nombre de los objetos que vamos a codificar con sus propiedades. Este documento se envía tanto a diseñadores como a desarrolladores, en este caso la misma persona, yo.
- Parte de diseño: El diseñador crea cómo será el aspecto de la aplicación usando los datos que le han pasado en el documento de wireframe.
- Parte de desarrollo: Usando el archivo proveniente de las partes de wireframe y diseño, el desarrollador crea la funcionalidad del programa en lenguajes del Framework .NET.
- Integración: Cuando las partes de diseño y desarrollo han terminado su trabajo se integran para tener el resultado final de la aplicación.

XAML es relativamente simple. Es un lenguaje de programación de propósito general perfecto para construir e inicializar objetos en .NET, quien incluye compilador y un parser de tiempo real para dicho lenguaje.

XAML se puede comparar con HTML, SVG (Scalable Vector Graphics) u otros lenguajes de dominio específico puesto que es un modo de usar las APIs de .NET.

XAML consiste en una serie de reglas que dicen cómo los parsers/compiladores deben tratar un archivo XML además de algunas palabras claves, pero no define elementos interesantes él mismo.

Sin embargo, el rol que XAML juega en relación con WPF es un poco confuso, así que la primera cosa de la que tenemos que tener clara es que WPF y XAML pueden ser usados independientemente. Aunque XAML fue originalmente diseñado para WPF se puede aplicar a otras tecnologías también (como el antes mencionado Windows Workflow Foundation). De hecho, debido a su naturaleza como lenguaje de propósito general, XAML puede ser aplicado a cualquier otra tecnología .NET.

Además, usar XAML con WPF es opcional porque todo lo hecho con XAML puede ser hecho con tu lenguaje .NET favorito como C# (nótese que lo contrario no se puede hacer, o sea, que todo lo que haces con un lenguaje .NET como C# no puedes hacerlo con XAML).

De cualquier modo por lo explicado en el apartado anterior y los beneficios que otorga, es muy raro ver una aplicación desarrollada en WPF sin haber usado XAML.

Elementos y atributos en XAML.

Las especificaciones de XAML definen reglas que asignan los namespaces, tipos, propiedades y eventos de .NET en namespaces, elementos y atributos de XML.

Hay que tener en cuenta que XAML también es un lenguaje sensible a las mayúsculas con lo cual hay que tener mucho cuidado a la hora de definir los elementos.

Para ver en profundidad como se representan estos elementos en XAML vamos a poner ejemplos y ver cómo serían sus homónimos en un lenguaje de programación como C#.

Para representar un botón estándar

Código en XAML:

```
<Button xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation Content="OK"/>
```

Código en C#:

```
System.Windows.Controls.Button b = new System.Windows.Controls.Button();
b.Content = "OK";
```

Fragmento de código 8. Diferencias entre botón estándar en XAML y en C#

Una ventaja que salta a la vista es que con XAML tenemos sólo una línea de código aunque parezca estar más cargada, mientras que con C# ya tenemos que crear dos, una para definir el botón en sí y otra para definir qué habrá en su interior.

Además, el código el escrito en XAML puede ser visto en Internet Explorer directamente como aparece en la imagen a continuación, mientras que el código C# debe ser compilado además de necesitar código extra, esta es una de las ventajas fundamentales para decantarse entre un lenguaje y otro.

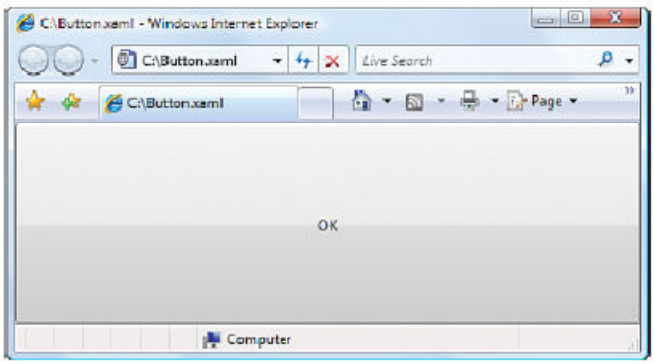


Ilustración 18. Botón estándar creado en XAML

Para representar un botón más original

A continuación quiero mostrar cómo se podrí crear otro botón en el que se puede apreciar un contenido visual mucho más rico que el anterior. Esta diferencia no entraña una gran cantidad de código, esa es una de las mejores ventajas de XAML. Añadiéndole un fondo a un efecto podemos conseguir resultados asombrosos.

Código en XAML:

```
<Button Width=”100”> OK
        <Button.Background>
                LightBlue
        </Button.Background>
</Button>
```

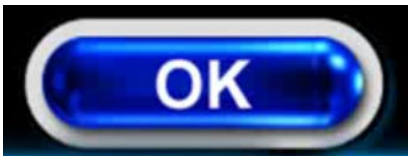


Ilustración 19. Botón con efectos

Código homónimo en C#:

```
Button b1 = new Button();
b1.Content = “OK”;
b1.Background = new
        SolidColourBrush(Colors.LightBlue);
b1.Width = 100;
```

Fragmento de código 9. Diferencias entre botón con efectos en XAML y C#

Para declarar un elemento de XML como si fuera XAML y que por tanto se conociera con el nombre de objeto, hay que instanciarlo por la vía del constructor por defecto en el framework .NET.

Para definir un atributo del objeto en XAML hay que definir una propiedad del mismo nombre (llamado atributo de la propiedad) o crear un controlador para un evento con el mismo nombre (llamado atributo de evento).

Por ejemplo, he aquí una actualización de los botones que habíamos creado anteriormente en los que sólo se había trabajado con una propiedad, “Content”, que permitía insertar el texto que el usuario vería dentro del botón. Ahora voy a implementar otra, el lanzador del evento (event handler), que supone el nombre del método que tendrá que ejecutar el programa una vez el usuario pinche encima del botón en cuestión.

De este modo puede verse una de las mejores facetas de XAML y WPF como es que separe completamente la parte del diseño de la parte de implementación puesto que el método al que se llama estará en un archivo diferente con extensión en mi caso .cs y que contendrá el código en C#. Además hay que tener en cuenta que aquí terminaría la función de XAML, por eso insisto en que es necesario trabajar con otros lenguajes además del mencionado.

Código en XAML:

```
<Button xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation>
Content="OK" Click="button_Click"/>
```

Código en C#:

```
System.Windows.Controls.Button b = new System.Windows.Controls.Button();
b.Click += new System.Windows.RoutedEventHandler(button_Click);
b.Content = "OK";
```

Fragmento de código 10. Diferencias de eventos en un botón para XAML y C#

Namespaces en XAML

Al comparar los fragmentos de código en XAML y C# el lector habrá podido darse cuenta de la siguiente línea de código donde llama la atención encontrar una dirección web:

```
<Button xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation>
```

Fragmento de código 11. Definición de un namespace en un objeto

Pues bien, es un elemento conocido como **namespace** en XAML (y XML) que al ser escrito automáticamente queda asignado al namespace de .NET "System.Windows.Controls".

Un objeto definido en XAML debe siempre especificar al menos un namespace XML que se use para definirse él mismo y a sus elementos hijo.

Al crear código en XAML puedes declarar namespaces adicionales en la propia ruta del objeto o en sus hijos, pero hay que tener en cuenta que cada uno debe tener un prefijo distinto para ser usado por alguno de los identificadores de ese namespace. Por ejemplo, muchos archivos WPF escritos en XAML suelen usar un segundo namespace al que declaran usando el prefijo x detrás de xmlns y al que quitan el último /presentation, de modo que quede así:

```
xmlns:x= http://schemas.microsoft.com/winfx/ 2006/xaml
```

Sin embargo y aunque no esté definido, de manera convencional se usa la siguiente notación para definir un namespace principal y uno secundario:

Namespace principal: xmlns= http://schemas.microsoft.com/winfx/2006/xaml/presentation
Namespace secundario: xmlns:x= http://schemas.microsoft.com/winfx/2006/xaml

Fragmento de código 12. Diferencia entre namespace principal y secundario en XAML

Donde desde luego es más fácil identificar el namespace principal como el que no tiene prefijo adicional y que encima acaba con /Presentation. El resto de los namespaces tendrán prefijos tales como x, y, z…

XAML tiene mucha gramática, es difícil y pesado de escribir.

Es otra de las características de los lenguajes de marcado como XML, HTML o XAML.

A nadie le gusta escribir muchas y largas líneas de código al programar y para ello se crearon las herramientas que trabajan con estos lenguajes. Estas se basan en los patrones de autocompletado de frases y en la gramática bien formada que la componen. La naturaleza transparente y bien especificada de XML hace que sea fácil e intuitivo usar este tipo de programas y que se puedan crear otros nuevos que ayuden a la programación y al depurado de código. Por citar ejemplos tenemos Expression Blend o Microsoft Frontpage.

Por otro lado uno puede pensar que este tipo de herramientas se han creado actualmente para satisfacer estas necesidades y solventar estos problemas, pero XAML ya fue creado con vistas a ser generado con herramientas de este tipo.

A pesar de esto creo que aprender XAML y estar familiarizado con ello es importante porque además se está aprendiendo XML. Además es importante conocer la diferencia entre esos lenguajes y los estructurados para introducirnos en el modo de programación más potente actualmente.

Conclusión

Ahora sí sabemos qué relación tiene XAML con WPF y lo que es más importante, tenemos la mayor parte de la información teórica sobre como traducir ejemplos XAML a lenguajes del marco de trabajo .NET 3.0 como C# y viceversa (aunque esto no se pueda dar siempre).

XAML se complementa muy bien con otros lenguajes, por ejemplo en el caso de expresiones como “null” donde es igual que en otros muchos.

Por tanto, usar un lenguaje de marcado como este, que es una evolución del mundialmente conocido XML, es un acierto para la creación de interfaces, por eso que lo uso en mi proyecto.

3.2.6. SQL

INTRODUCCIÓN

SQL (Structured Query Language) es el lenguaje universal usado para comunicarse con una base de datos. Hablamos por tanto de un lenguaje normalizado que nos permite trabajar en combinación con cualquier tipo de lenguaje (ASP, C# o PHP entre otros) y con cualquier tipo de base de datos (MS Access, SQL Server, MySQL...).

Pero no hay que confundirse, el hecho de que sea estándar no quiere decir que sea idéntico para cada base de datos. La base de la gramática se conserva en todas las versiones (sentencias como “Create”, “Delete”, “Insert”, “Drop”, “Update” o “Select”) que se usan para crear tablas o campos, eliminar,

insertar, borrar, actualizar o seleccionar son iguales en todas. Sin embargo hay determinadas bases de datos que implementan funciones específicas que no tienen necesariamente que funcionar en otras.

Algunos de los sistemas más famosos que usan SQL son Oracle, Microsoft SQL Service, Access o Sybase.

Sus principales características, que suponen sus ventajas, son las siguientes

• SQL está basado en un marco teórico sólido como es el álgebra relacional.

• Simplifica al máximo los conceptos. Se trata de bases de datos con tablas, por tanto, ¿Qué hay más fácil que trabajar con las filas y las columnas?

• Es fácil y rápido aprendizaje

• Se integra fácilmente con cualquier lenguaje de programación

• Es un estándar en el tratado de las bases de datos.

• Posee una arquitectura cliente-servidor muy sólida. (SQLcourse.com)

SQL EN PROFUNDIDAD

Puesto que SQL es un lenguaje que considero de vital importancia quiero explicar a este nivel los detalles de implementación y el significado del código. De este modo cualquier lector que conozca cualquier lenguaje de programación va a poder aplicar las sentencias aquí escritas en su código. Además considero que saber hacer las operaciones más básicas es muy importante, de ahí que las incluya en este estado del arte. (Desarrolloweb.SQL)

Una base de datos está compuesta de tablas y estas tablas tendrán:

- Filas: Serán los registros.
- Columnas: Serán las características de los registros.

Estas características están ligadas a su tipo de datos, por eso es de vital importancia la elección del mejor posible en función de los valores que se vayan a dar. Imagine una tabla con millones de registros como la que se puede dar en una gran empresa y que uno de los campos ocupe demasiado. Ocurrirá que al repetirse tantas veces se puede llegar a perder gigas de memoria física.

Como he dicho antes, todas las bases de datos tienen una parte en común. En concreto, la parte de la definición de los tipos de dato contiene a los más importantes que muestro a continuación:

Alfanuméricos	Pueden contener cifras y letras. Longitud limitada de 255 caracteres.
Numéricos	Existen de varios tipos aunque principalmente se usan enteros (sin decimales) y reales (con decimales).
Booleanos	Pueden tener los valores Verdadero y falso (TRUE o FALSE)
Fechas	Almacenan datos con tal formato a elegir por el usuario. En el caso de la mía uso DD-MM-AAAA.
Autoincrementables	Son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Se usan como identificadores de registros o como contadores.

Tabla 6. Definición de tipos de datos más comunes en SQL

Más concretamente en SQL los datos pueden ser de los siguientes tipos:

INT o INTEGER: Almacena números enteros mediante un campo de 4 Bytes.

TEXT: Almacena una cadena de caracteres alfanuméricos.

DOUBLE o REAL: Almacena números reales grandes que pueden tener decimales con un campo de 8 Bytes.

CHAR: Ocupa 1 byte por carácter que almacena. La longitud de este campo viene predefinida.

VARCHAR: Almacena caracteres alfanuméricos de longitud variable.

DATE: Almacena una fecha. Existen varios formatos dependiendo de la base de datos. Usa 3 Bytes.

BIT o BOOLEAN: Almacena un bit de información que puede ser 0 o 1.

Operaciones principales en SQL:

La mayoría de las bases de datos tienen asociadas potentes editores que generan el código como SQLServer, Access, SQLite (mi caso) o MySQL. Sin embargo al estar los datos en un servidor a menudo puede ser necesario acceder manualmente, por eso voy a explicar un poco el código para hacerlo.

1. Creación de tablas:

Es muy sencillo, se definen los tipos de campos principales y los índices con esta sintaxis:

```
Create Table nombre_tabla
(
        nombre_campo_1 tipo_1 restricciones
        nombre_campo_2 tipo_2 restricciones
        nombre_campo_n tipo_n restricciones
        Key(campo_x,...)
)
```

Un ejemplo sería la tabla cliente.

```
Create Table clientes

(

        Id_cliente INT(3) NOT_NULL AUTO_INCREMENT

KEY (id_cliente)

)
```

Fragmento de código 13. Creación de una tabla en SQL

Donde nombre_campo sería id_cliente, el tipo de datos sería un entero (INT) de 4 bits y las restricciones que tendría es que no puede tener un valor nulo y que ese campo se autoincrementará según se vayan añadiendo en la tabla indicando lo que sería un campo clave de ordenación en ella.

Con respecto a los índices se está diciendo que la clave de esta tabla será el campo id_cliente, o sea, que no habrá dos clientes con el mismo id.

2. Introducción de registros en una tabla:

Los registros se introducen con sentencias que emplean la instrucción “INSERT” mediante esta sintaxis.

```
Insert Into nombre_tabla (nombre_campo1, nombre_campo2,...)

Values (valor_campo1, valor_campo2...)
```

Un ejemplo sería:

```
Insert Into clientes (nombre, pedidos, id_cliente)

Values ('Vicente', 2, 099)
```

Fragmento de código 14. Introducción de registros en una tabla en SQL

No es necesario escribir un valor para todos los campos, pero en algunas ocasiones puede serlo si así lo indican las restricciones. En caso de no hacerlo los valores asignados serán los dichos por defecto y en caso de que estos no se hayan definido, se les asignará el valor "NULL".

3. Borrar registros en una tabla.

Para realizar esta acción hemos de utilizar sentencias que emplean la instrucción "DELETE" mediante esta sintaxis.

```
Delete From nombre_tabla Where condiciones_seleccion
```

Para esta operación debemos hacer una selección de lo que queremos borrar con la cláusula "WHERE", de modo que si quisiéramos borrar todos los registros de la tabla clientes cuyo nombre fuera "Vicente" sería de este modo:

```
Delete From clientes Where nombre='Vicente'
```

En esta operación tenemos que tener mucho cuidado ya que si no añadimos la cláusula WHERE podemos borrar todo el contenido de la tabla como ocurriría en este ejemplo:

```
Delete From clientes
```

Fragmento de código 15. Borrado de registros en una tabla en SQL

4. Editar registros en una tabla.

Para realizar esta acción usamos la instrucción "DELETE" pero como ocurría en el caso de borrado tenemos que especificar una condición con "WHERE". Además utilizamos "SET" para especificar cómo deseamos modificarlo.

La sintaxis sería así:

```
Update nombre_tabla Set nombre_campo1 = valor_campo1, nombre_campo2 = valor_campo2,...
Where condiciones_de_selección
```

Un ejemplo que sustituiría todos los campos con nombre "Vicente" a campos con nombre "Papa Bear" sería:

```
Update clientes Set nombre = 'Papa Bear' Where nombre=`Vicente`
```

Como ocurría con el borrado, si queremos editar todos los registros de la tabla escribiríamos:

```
Update clientes Set nombre='Papa Bear'
```

Fragmento de código 16. Modificación de registros en una tabla en SQL

Entonces todos los clientes tendrían de valor del atributo nombre, "Papa Bear".

5. Hacer consultas sobre las tablas.

Se utiliza la instrucción "SELECT". Hay que especificar:

Los campos a seleccionar.

La tabla en la que están.

De esta manera y con esta sintaxis:

```
Select campo1,campo2 From nombre_tabla
```

Si quisiéramos seleccionar todos los campos de la tabla utilizaríamos el comodín *

```
Select * From nombre_tabla
```

Fragmento de código 17. Consulta básica en SQL

Sin embargo en la actualidad resulta muy interesante el hecho de poder filtrar búsquedas ya que no siempre vamos a poder definírselo de manera implícita como os he mostrado anteriormente.

Para ello utilizamos cláusulas como

- "WHERE LIKE": `Select * From clientes Where nombre Like 'Vicente'`

Fragmento de código 18. Consulta en SQL usando "Where Like"

Donde seleccionaríamos de la tabla clientes a aquellos que se llamen Vicente.

- "ORDER BY": Ordena los resultados de la consulta en función de uno o varios campos.

```
Select * From clientes Where nombre Like 'Vicente' Order By pedidos
```

Fragmento de código 19. Consulta en SQL usando "Oder By"

Donde seleccionaría los clientes 'Vicente` y mostraría los resultados ordenado por el número de pedidos que ha hecho. También se pueden poner varios campos, lo que daría como resultado a un orden por el primero de ellos y en caso de que haya dos iguales se ordenarán por el segundo campo.

- "DESC": Ordena los resultados de la consulta de forma descendente.

```
Select * From clientes Order By pedidos Desc
```

Fragmento de código 20. Consulta en SQL usando "Desc"

- "DISTINCT": Para hacer selecciones sin coincidencia, es decir, que a la hora de buscar registros no se nos repitan los que cumplan la condición.

```
Select Distinct población From clientes Order By población
```

Fragmento de código 21. Consulta en SQL usando Distinct

Además de cláusulas para filtrar las selecciones utilizamos **operadores** tales como >, <, >=, <=, <> (distinto), =, And, Or, Not, Like, In, Not In, Is Null, Is Not Null, Between, Distinct, Desc, *, % (sustituye cualquier cosa dentro de una cadena), _ (sustituye un carácter dentro de una cadena).

Para concluir este apartado quisiera escribir un par de ejemplos en el que uso operadores y cláusulas a la hora de hacer consultas.

```
Select * From clientes Where pedidos Like 2 And Not nombre Like 'Vicente'
```

Selecciona de la tabla clientes aquellos cuyo número de pedidos sea dos y que no se llamen "Vicente".

```
Select * From clientes Where nombre like 'V%' And pedidos Between 2 And 10
```

Selecciona de la table clientes aquellos cuyo nombre empieza por V y su número de pedidos esté entre 2 y 10

```
Select * From clientes Where nombre In ('Vicente','Alejandro ','Juan Carlos')
```

Selecciona de la tabla clientes aquellos cuyo nombre sea Vicente, Alejandro o Juan Carlos.

Fragmento de código 22. Ejemplo de consultas en SQL

Funciones principales en SQL

SQL permite utilizar funciones predefinidas que ayudan a expresar selecciones. A continuación voy a mostrar las más utilizadas.

Sum(campo): Calcula la suma de los registros del campo especificado.

Avg(campo): Calcula la media de los registros del campo especificado.

Count(*): Nos da el valor de los registros que han sido seleccionados.

Max/Min(Campo): Nos indica el valor máximo o mínimo del campo.

3.3. HERRAMIENTAS UTILIZADAS.

En esta sección voy a introducir al lector en las herramientas con las que he trabajado para desarrollar el software que compone este proyecto.

3.3.1. MICROSOFT VISUAL STUDIO 2008.

Ilustración 20. Logo de Visual Studio (FreeSoftware30)

Microsoft Visual Studio.NET 2008 es la penúltima versión de su entorno de desarrollo en la plataforma .NET. Utilizo esta versión porque era la que pude descargarme con Microsoft MSDN y mi licencia de estudiante.

Visual Studio ofrece una interfaz común para trabajar de manera cómoda y visual con cualquiera de los lenguajes de la plataforma .NET como son C++, C#, Visual Basic.NET y JScript.NET, aunque se van añadiendo nuevos lenguajes mediante los plugins que proporcionan sus fabricantes.

Visual Studio permite a los desarrolladores crear aplicaciones o servicios web en cualquier entorno que soporte la plataforma .NET a partir de su versión de 2002, así como aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. (Studio)

Historia

A lo largo del tiempo se han ido creando nuevas versiones o actualizaciones que han ido desbancando a sus anteriores versiones. A continuación muestro una lista con las más importantes, las que pasaron a la historia por sus ventajas o desventajas donde además de situarlas en el tiempo, resumo brevemente sus características para que el lector pueda conocer como paralelamente ha ido evolucionando al framework .NET. (Desarrolloweb.VisualStudio),

Visual Studio 6.0

Se lanzó en 1998 y produjo que todas las versiones existentes del framework .NET pasaran a ser la versión 6.0 en vez de la 1.0 o 1.1 como ocurría.

Esta versión se conoce porque fue la última versión en incluir Visual J++ ya que el hecho de hacerlo le supuso problemas legales con Sun Microsystem (Creador de Java), quienes finalmente denegaron legalmente el permiso a Microsoft para crear aplicaciones que usaran su máquina virtual Java.

Además, Visual Studio 6.0 fue la última versión en que Visual Basic se incluía de la forma en que se conocía hasta entonces, luego pasó a ser Visual Basic .NET.

De ese modo la versión 6.0 pasó a la historia ya no por unificar todas las tecnologías, si no porque fue cuando Microsoft perdió un importante lenguaje como era Visual J++.

La interfaz de este programa tenía este aspecto:

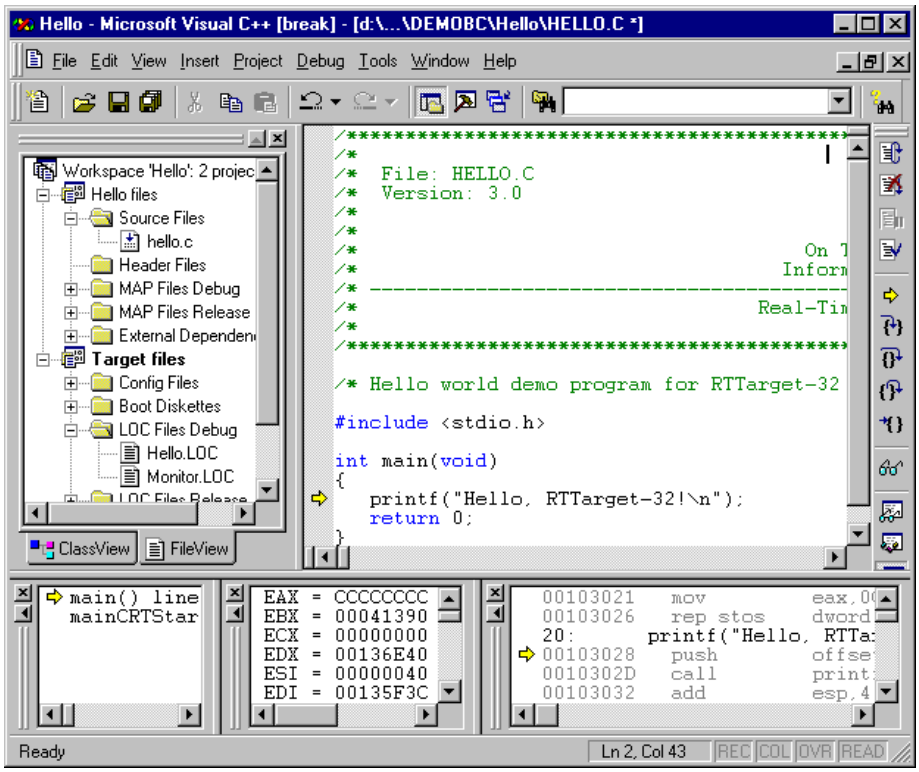


Ilustración 21. Interfaz de Visual Studio 6.0 (testech-elect)

Visual Studio .NET (2002)

Fue la primera versión a la que se le incluyó el nombre .NET puesto que ya no se compilaba en lenguaje máquina si no en el lenguaje intermedio del que he hablado anteriormente, MSIL, de modo que se había creado una plataforma multilenguaje con el fin de aunar todo bajo el mismo nombre.

Otra gran novedad fue la inclusión por primera vez el lenguaje C# basándose como he explicado anteriormente en C++ y Java.

Además, Visual Studio .NET se orientó a crear programas basados en Windows con Windows Forms en vez de COM, aplicaciones y sitios web y dispositivos móviles puesto que esos mercados se estaban abriendo a la gran cantidad de tecnologías emergentes.

En temas de interfaz supuso un gran cambio, la versión que utilizo es bastante parecida a ella.

Tenía este aspecto:

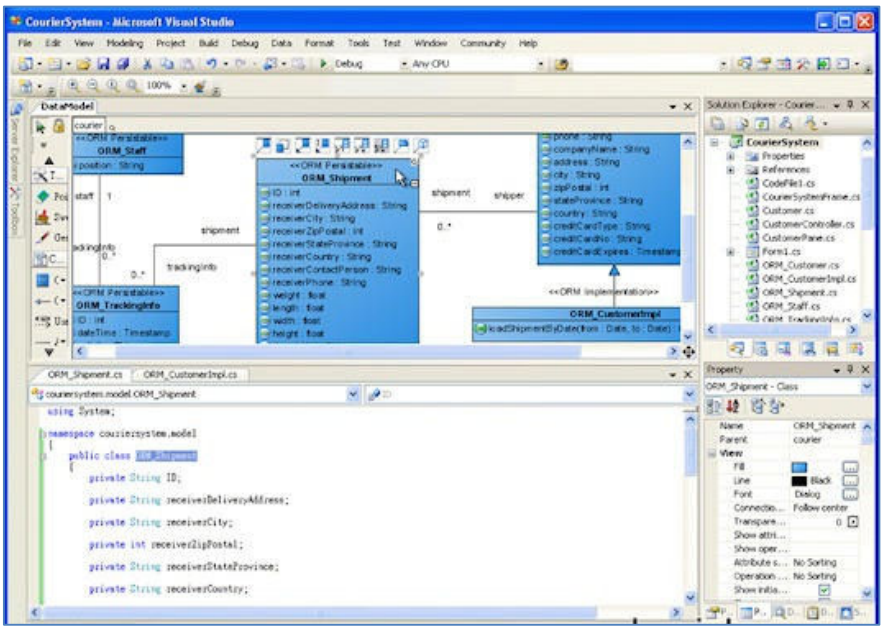


Ilustración 22. Interfaz de Visual Studio .NET 2002 (malavida.com)

Visual Studio .NET 2003

Visual Studio .NET 2003 supone una ligera actualización del Framework de su versión 1.0 a su versión 1.1. Se añade soporte sobre todo en plataformas móviles.

Esta versión de .NET duró sólo dos años hasta que salió la versión 2005, sin embargo, al ser la de 2003 una versión muy utilizada, Microsoft se vio obligado a lanzar en 2006 un Service Pack para corregir errores existentes.

Visual Studio 2005

En 2005 le volvieron a quitar el distintivo .NET al nombre.

Se actualizó la versión del Framework al 2.0.

Se añadieron tipos genéricos, lo que supuso el hecho de encontrar muchos más errores en la compilación en vez de en la ejecución.

Con esta versión además se introdujo soporte para tecnologías de 64 bits así como compiladores y librerías de 64 bits, lo que empezaba a presagiar que las tecnologías de 32 bits podrían quedarse obsoletas en un tiempo que todavía no ha llegado.

También se incluye un entorno para publicación web y pruebas de carga para comprobar el rendimiento de los programas bajo varias condiciones de carga.

Además con esta versión 2005 se empezó a dar importancia a las ediciones Express de Visual Studio diseñadas para principiantes, aficionados y pequeños negocios de distribución gratuita. Con esto ganaron más usuarios que sin duda han servido para lanzar Visual Studio y la plataforma .NET hasta el día de hoy.

Visual Studio 2008

Con esta versión nació el actual Framework .NET 3.5.

Se implementó para aprovechar los avances tecnológicos del nuevo sistema operativo de Microsoft, "Windows Vista", que dicho sea de paso ha sido considerado un gran error por parte de los usuarios porque daba una gran inestabilidad a los sistemas. Prueba de ello es la rápida subsanación de errores con la creación del último sistema operativo, Windows 7, que repara muchos de los errores pero utiliza novedosas tecnologías que ya incluía su predecesor.

Relacionado con el trabajo sobre las bases de datos se incluye LINQ, un nuevo conjunto de herramientas diseñadas para reducir la complejidad al acceder a las bases de datos con SQL.

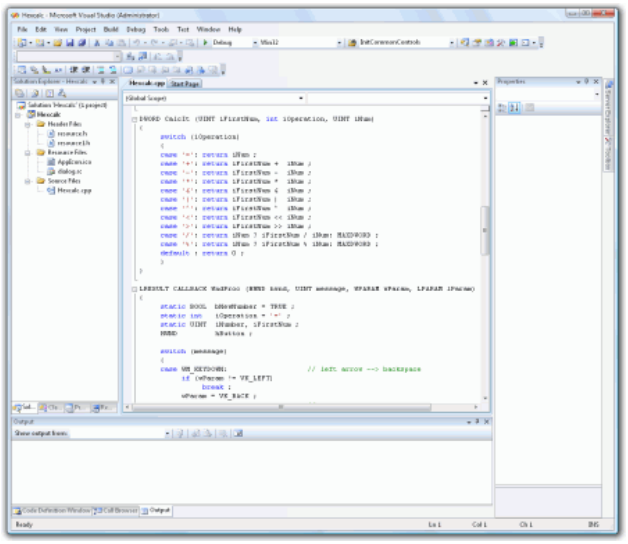


Ilustración 23. Interfaz de Visual Studio 2008 (TomsHardware)

Visual Studio 2010

Con esta versión se actualiza el Framework a .NET 4.0, su última versión hasta la fecha. En ella se mejora mucho el trabajo con las bases de datos, sobre todo con Oracle e IBM DB2.

Se ha mejorado mucho Visual Studio para trabajar con tecnología como WPF y WCF.

Además se logra una mayor integración con Microsoft Office y elementos multimedia como música, fotos o listas de reproducción.

Su interfaz tiene este aspecto.

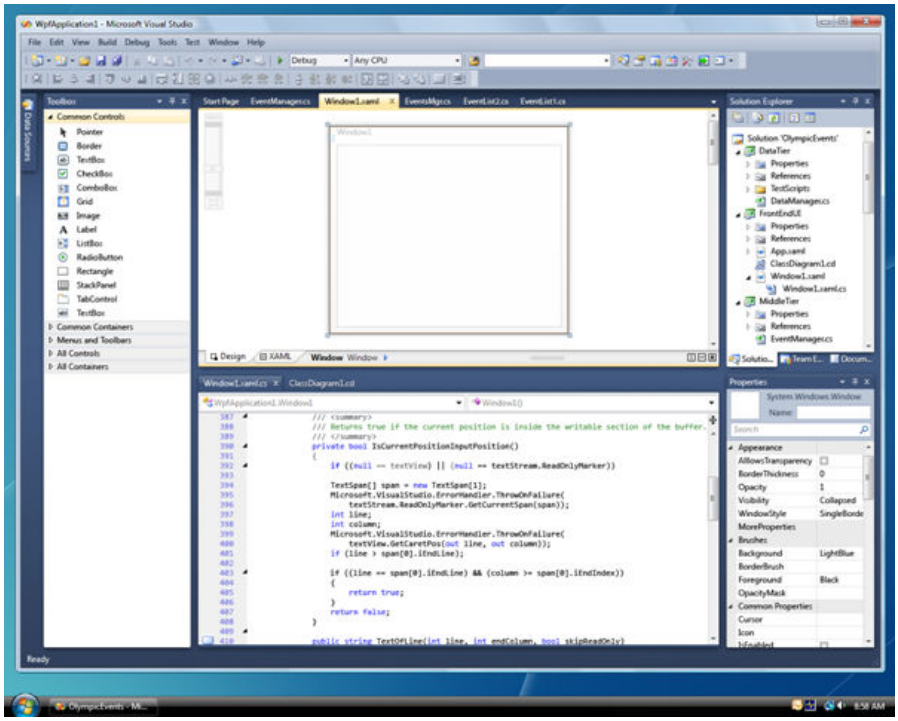


Ilustración 24. Interfaz de Visual Studio 2010 (MSDN, Blogs)

3.3.2. MICROSOFT EXPRESSION BLEND 3.

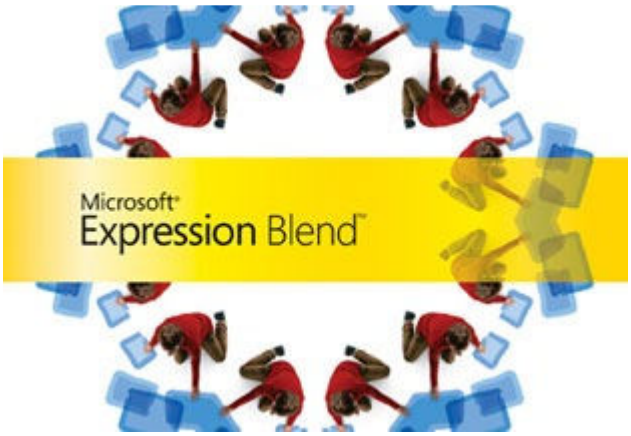


Ilustración 25. Logo de Expression Blend (Blogspot.com)

Expression Blend es una herramienta que distribuye Microsoft y que perteneciente al paquete de aplicaciones Expression Studio de Microsoft que sirve para integrar al diseñador gráfico en el desarrollo de aplicaciones sin tener que obligarles a escribir código especializado.

Sirve para trabajar con WPF y supone una mejor expresión de los diseñadores que con un Windows Form. Mientras se crea la interfaz se irá generando un código que será común con el de los desarrolladores, no habiendo diferencia por tanto entre lo que unos diseñan y lo que al final se desarrolla.

Sin embargo esta herramienta, nacida como una auxiliar de Visual Studio, no habría triunfado si no permitiera a sus proyectos integrarse perfectamente con los de esta última. De este modo los diseñadores pueden pasarle el proyecto creado con Expression Blend a los desarrolladores y estos abrirlos con Visual Studio, entorno donde están acostumbrados a programar.

En el caso de este proyecto ha supuesto una rapidez en el desarrollo ya que he podido separar dos partes tan importantes como el diseño y la codificación. Para trabajar con WPF es muy útil porque al crear el diseño de tu interfaz irá generando un código XAML por detrás que usaré con Visual Studio posteriormente.

Para diseñar es más potente que Visual Studio y al estar centrado sólo en esta parte es mucho más intuitivo y simple, no tienes que preocuparte de la infinidad de botones y funciones que Visual Studio ofrece, sólo centrarte en que la aplicación quede bonita y sea estable.

Es lógico pensar que los grandes desarrolladores pueden no ser muy buenos diseñando, por eso ha nacido este software, ante la necesidad de aunar estos dos mundos que trabajaban separados por un mismo propósito. Por eso Microsoft decidió crear la primera herramienta 100 % para Diseñadores perfectamente compatible con los proyectos de Visual Studio. Ahora se puede crear interfaces de asombrosa calidad final, con efectos 3D, todo un manejo Multimedia y completamente pasible de eventos. (elChelo)

¿Cuál es la relación entre el Expression Blend 3 y .NET Framework?

Tanto .NET Framework 3.0, como 3.5 y 4.0 usan la plataforma WPF con lenguaje XAML. Es la parte en común, Expression Blend genera un código en este lenguaje que es válido tanto para WPF como para otras tecnologías de las que he hablado anteriormente.

Expression Blend incorporta SketchFlow, que es una herramienta con la que podemos sin escribir una línea de código, presentar de manera sencilla la visión de una aplicación o qué comportamientos o animaciones tendrá.

Está diseñada para generar ese primer archivo que el diseñador realiza a modo de boceto y que enseña a la empresa para la que está trabajando buscando su aprobación.
A continuación voy a mostrar un ejemplo de cómo quedaría la visión de una aplicación con esta herramienta.

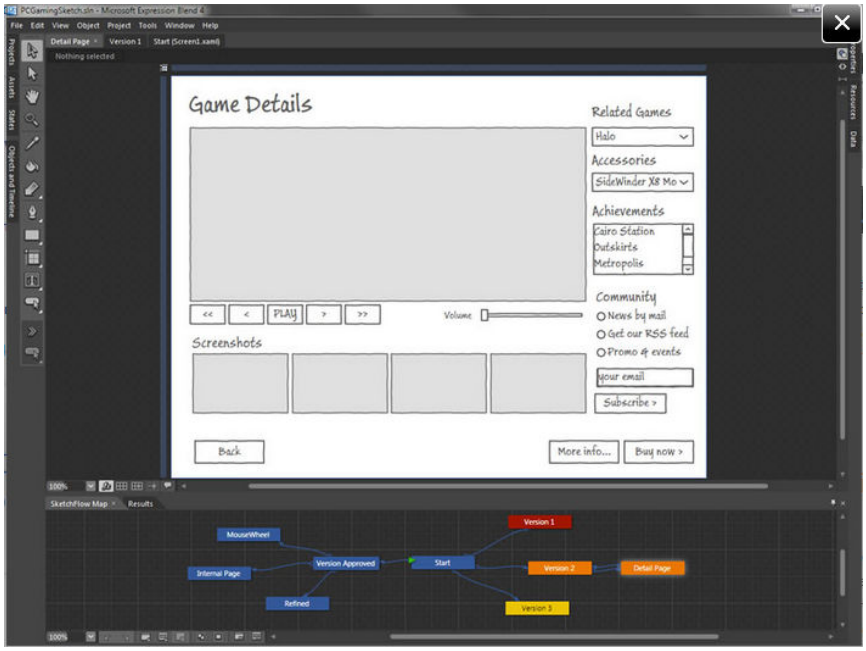


Ilustración 26. Ejemplo de pantalla realizada con SketchFlow (Wpdevelopers)

La interfaz de Expression Blend

Dado que el diseño en WPF tiene muchísimas posibilidades, Microsoft ha querido sacarle provecho creando una interfaz que pone al alcance de la mano del diseñador herramientas que eran más difíciles de encontrar en Visual Studio, por ejemplo. Se puede decir que hace más fácil diseñar una aplicación.

La interfaz de Expression Blend 3 tiene este aspecto:

Ilustración 27. Interfaz de Expression Blend 3 (Malavida.Blend)

Expression Blend y este proyecto

He usado Expression Blend para crear la interfaz de la aplicación que he realizado.

Tras empezar a usar Visual Studio me resultaba difícil aprender a utilizar otro programa aun habiendo leído las maravillosas virtudes de este programa. Sin embargo al instalarlo no tuve ningún problema para aprender a utilizarlo, todo era más fácil que con Visual Studio, los menús más claros, te expone mejor las inmensas posibilidades que tiene para crear efectos superiores. Creo que su éxito tiene que ver con que está más orientado al diseño y por tanto ofrece una interfaz mucho más gráfica, mejor ordenada y herramientas más accesibles que del otro modo.

Creo que es muy importante separar el diseño y la programación, por eso en ese aspecto Visual Studio pierde mucho, es un programa tan complejo que a la hora de centrarse en cosas como el diseño deja mucho que desear, por eso se crearon este tipo de aplicaciones como Expression Blend.

El trabajo con este programa ha sido de lo más agradable y en ningún momento he tenido problemas para integrar los proyectos con Visual Studio. (Expression Microsoft)

3.3.3. MICROSOFT VISIO PROFESSIONAL 2007.

Es un software de dibujo vectorial para Microsoft Windows.

No lo he usado demasiado en este proyecto más que para ayudarme a hacer gráficos y esquemas para la memoria. Sí lo usé más en el proyecto que se me asignó previamente para generar los diagramas de clases, por eso no voy a centrarme mucho en él.

Esta herramienta permite realizar diagramas de muchos tipos para oficinas y proyectos tales como bases de datos, de flujo de programas, UML… entre otros. (Microsoft.Office)

3.3.4. SQLITE Y SQLITE ADMINISTRATOR

SQLite es una librería implementada en código libre para el motor de la base de datos SQL.

A diferencia de las otras no tiene un servidor distinto, sino que lee y escribe directamente sobre archivos del disco de modo que una base de datos de SQL con todos sus componentes (triggers, tablas…) quedan contenidos en un solo archivo, lo que hace que sea muy cómodo trabajar con él.

SQLite es una librería compacta. A pesar de tener todas las características que la hacen potente, puede llegar a ocupar 300 Kb (aunque depende del compilador). Sin embargo al quitar algunas de las funciones opcionales podemos llegar a tener que el tamaño de esta librería es de apenas 180 Kb. Además, SQLite puede ser configurado para ejecutarse ocupando un tamaño mínimo, característica perfecta si se quiere trabajar con teléfonos, PDAs o reproductores MP3 entre otros. Sin embargo hay que destacar que la rapidez con la que trabaja SQLite viene a menudo ligada al espacio que asignas para él, lo que no necesariamente quiere decir que al ocupar poco vaya mal.

SQLite al tener código libre ha sido cuidadosamente testada por miles de usuarios y se ha ganado la reputación de ser una biblioteca muy fiable. La mayoría del código fuente de SQLite está escrito sólo para poder comprobar y verificar ejecutando millones de casos con fallos de sistema. Y no nos confundamos, SQLite tiene fallos, pero lo que le hace diferente del resto de los competidores (comercializables o no) es que actualiza la lista verdadera de problemas que se dan a tiempo real, no se reserva el derecho de no hacerlo o de omitir información. Además por si cupiera alguna duda el código de SQLite está comprobado por un equipo internacional de desarrolladores que se dedica a esto a tiempo completo. De modo que el código de SQLite es libre, si, pero hay un soporte profesional disponible al alcance del usuario. (Sqlite)

Otras características de SQLite

- Es transaccional, o sea, que todos los cambios y consultas son atómicas, consistentes y perdurables. SQLite realiza todas las operaciones aunque el sistema operativo interrumpa la operación o haya un fallo de corriente, por ejemplo.
- SQLite no tiene que ser instalada antes de usarse. Tampoco hay que iniciar, pausar o detener ningún proceso o que un administrador cree una nueva instancia de base de datos o que asigne permisos a los usuarios. No hay que decirle al ordenador que SQLite está trabajando y no usa archivos de configuración, simplemente funciona de manera simple y clara para el usuario.
- Soporta e implementa la mayoría de las características de SQL92 a excepción de unas pocas.
- Soporta un gran tamaño de bases de datos del orden de Terabytes así como cadenas de caracteres de gran tamaño (Gigas).
- SQLite funciona solo, no depende de ningún otro programa.
- Es fácil de portar a otros sistemas, no solo funciona en Windows, Linux o Mac OS X.
- Es perfectamente compatible con las bases de datos que se usan en MNC (Access)

Todas estas características supusieron una ventaja en la elección de la base de datos para mi proyecto, sin embargo me llamó la atención saber que el formato de esta base de datos es independiente de la plataforma donde se genera, de modo que podemos crearla en un sistema de 32 bits que sabemos con certeza que funcionará en otro de 64 bits.

Esto es muy importante ya que la base de datos que utiliza mi aplicación y la de Alejandro Alonso será la misma, siendo nuestros sistemas operativos de 64 y 32 bits respectivamente.,

Para administrar las bases de datos nos hemos basado en SQLite pero realmente hemos utilizado un programa llamado SQLite administrator que pudimos adquirir gratuitamente. Nos ha servido de mucha ayuda ya que no hemos necesitado instalarlo y contiene una interfaz fácil e intuitiva que muestro con la siguiente imagen, que corresponde a la herramienta mostrando cómo haría una consulta (SELECT * FROM COMPUTER_INFO ORDER BY DATE DESC).

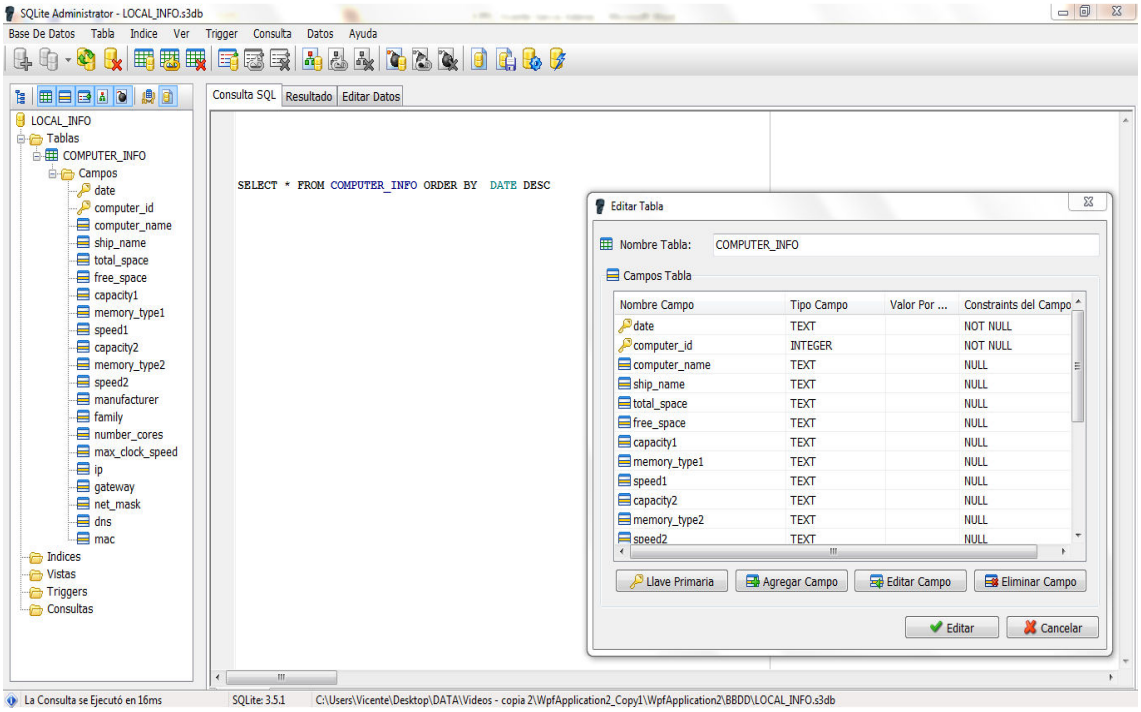


Ilustración 28. Interfaz de SQLite administrator en mi aplicación

4. ¿QUIÉN SERÁ EL USUARIO FINAL DEL PROGRAMA?

Este programa pretende ser instalado en el puente de mando del barco que contrate los servicios de MNC.

Esta especial zona del barco está dirigida por un capitán o un oficial además de una parte selectiva con los altos rangos de la tripulación. Son gente formada en la mar y todo lo que la relaciona, por tanto he de dar por supuesto tal y como me han confirmado en MNC, que su conocimiento en informática es muy limitado o nulo.

Este software en principio y bajo confirmación del director de MNC, está creado para tener cinco ordenadores conectados en red como ofrece en su paquete básico, pero dado que las bases para cinco ya están sentadas será muy fácil ampliarlo posteriormente a cuantos se quiera, de este modo podemos tener dos grandes tipos de usuarios finales:

- Yates y barcos particulares. En principio esta aplicación estaría destinada a este tipo de usuario con un máximo de 5 ordenadores en la misma red. Suelen ser barcos personales o de pequeñas empresas que tienen que viajar o simplemente que quieren estar de ocio sin el riesgo de perder los datos que llevan con ellos. Actualmente existen embarcaciones de este tipo que han contratado los servicios de MNC.
- Grandes barcos. Ampliando esta aplicación podría ser usada con tantos ordenadores como se quisiera, de modo que grandes embarcaciones tales como ferris y barcos de transporte podrían usarlo sin problema puesto que el modus operandi de la aplicación es muy simple y estable, lo que no daría problemas con una gran cantidad de ordenadores.

Partiendo de esta base, propuse a MNC desde el principio el crear un programa sencillo, casi para niños, que no requiera formación previa y lo más intuitivo posible. Que se juegue con los colores y sus significados además de con unas ventanas de ayudas con toda la información posible.

5. INTRODUCCIÓN AL DESARROLLO APLICADO.

En este capítulo quiero mostrar a gran escala cuales han sido los pasos que he dado en este proyecto sin entrar en detalle, cosa que haré más adelante. De modo que la persona que lo lea podrá tener una idea general clara de las partes en las que se ha divido el desarrollo de mi aplicación.

Cada una de estas tareas ha llevado un determinado tiempo pero este tipo de detalles serán explicados más adelante en la sección de planificación del proyecto.

Quiero que el lector tenga siempre presente que este proyecto se ha dividido principalmente en dos partes con las que al principio he trabajado en separado para finalmente unir hasta obtener la aplicación final, como son:

- Creación del programa “Touchable Screen Monitor Application” usando la tecnología WPF y los lenguajes C#.NET, XAML y SQL.
- Creación de la funcionalidad del programa de MNC con el plugin “Detailed Info Plugin”, escrito en lenguaje C#.NET.

Aunando estas dos voy son a grandes rasgos las más importantes, voy a detallar las demás fases lógicas que han supuesto el desarrollo de mi proyecto.

1. Familiarización con el entorno .NET.

Siempre había oído hablar de .NET pero la verdad es que apenas sabía de qué trataba. Para empezar a trabajar con él obtuve bibliografía que llevé a Finlandia para ir introduciéndome en este mundo. Aprendí qué es, qué tecnologías usa y de qué manera era positivo usarlo en mi proyecto, aparte de por el claro hecho de que es el marco de trabajo de la empresa para la que iba a desarrollar este software.

La bibliografía que he usado me ha acompañado durante todo este proyecto pues he tenido que referirme a ella constantemente para resolver las dudas pertinentes.

2. Lectura y aprendizaje de C#.

Con el tiempo aprendí a usar lenguajes de programación estructurada sobre todo, de manera que lo más parecido a C# que pude aprender fue el lenguaje C. Con ello aprendí a organizar más la mente de cara a la programación orientada a objetos pero sin duda el salto a un lenguaje como C# ha supuesto para mí un antes y un después en la informática ya que este es el lenguaje en el que se basa el Framework .NET, uno de los marcos de trabajo más potente hoy en día. Además, el empezar a usar este lenguaje y estas tecnologías supuso el salto de lo que estudiábamos en la carrera para formarnos y lo que las empresas y el mundo laboral realmente utilizan.

3. Aprendizaje de WPF.

Aprendí a utilizar WPF.NET gracias a programas como Visual Studio y Microsoft Expression Blend. Precisamente el hecho de empezar a utilizar este último programa supuso para mí un salto de calidad en el diseño para mi aplicación. Sus intuitivos menús y la extensa documentación que pude encontrar en internet (Gracias en especial a unos video tutoriales creados por Ivana Tilca, miembro de la comunidad MSDN) me brindó la oportunidad de aprender a usarlo sin demasiados problemas. Con ello my con el director del proyecto Patrick Sjöberg senté las bases del aspecto de la aplicación, para luego pasar a implementarla finalmente cuando la funcionalidad estuviera definida.

4. Aprendizaje de C#.

Tras conocer cómo iba a lucir mi programa físicamente dediqué un tiempo a formarme prácticamente en el lenguaje C#.NET para poder desarrollar la aplicación.

5. Desarrollo del plugin "Detailed Info".

Tanto Alejandro Alonso como yo teníamos que conseguir que un ordenador, mediante un plugin que creáramos nosotros nos proporcionara un archivo con sus datos para así poder tratarlos y mostrarlos.

Creamos el plugin que finalmente hiciera lo siguiente:

- Se ejecuta en el ordenador donde esté instalado.
- Genera un archivo XML con la información del mismo.
- Manda dicha información al servidor local del barco, quien lo inserta en una base de datos.

6. Integración de los datos de los ordenadores con una base de datos.

Una vez tenidos esos archivos XML con los que se iba a trabajar se nos presentaban dos opciones:

- Que los datos que queríamos leer provinieran directamente de tales archivos XML, con lo cual tendríamos, al mandarlos al servidor, una gran cantidad de éstos en el mismo.
- Que esos datos, al mandarlos al servidor, traspasaran su información a una base de datos creada sólo para ese propósito, donde quedarían guardados de manera ordenada. Esta base de datos estará emplazada físicamente en el servidor local del barco donde está nuestro pequeño sistema de ordenadores a monitorizar.

Al principio creíamos que podíamos utilizar la primera opción pero al ejecutar nuestro programa unas cuantas veces nos dimos cuenta de que era engorroso tener que trabajar con unos cuantos archivos con las dudas que estos suponían de cuándo habían sido creados, por qué máquina, etc. de manera que no quisimos imaginar la confusión que se generaría en un gran sistema usado cientos de veces. Por eso decidimos utilizar la segunda opción. Era más costoso para nosotros tener que crear una base de datos y formarnos para ello pero sería infinitamente más cómodo para el usuario y futuros desarrolladores.

De modo que la creamos con SQLite y modificamos nuestro programa principal de modo que además de crear esos datos, tras mandarlos al servidor, los integrara en la base de datos creada. Así conseguíamos

tener la información almacenada de manera ordenada, es fácil acceder y no tenemos que preocuparnos de trabajar directamente con varios archivos XML.

7. Recuperación de la información por parte de la aplicación en la pantalla táctil.

Una vez estaba guardada la información en la base de datos del servidor local, de la aplicación de la pantalla táctil en WPF.NET había que hacer consultas que nos la mostraran.

En función de esta información había que cambiar cómo reaccionarían los elementos de mi aplicación tales como los colores, tablas, etc. que hacen de esta interfaz una especialmente fácil e intuitiva.

8. Finalización de la interfaz.

Tras tener un boceto muy aproximado a la realidad y toda la funcionalidad de mi aplicación completada me dediqué a trabajar más con la interfaz para que no diera problemas y funcionara perfectamente ya con detalles como la resolución, el renderizado, etc.

9. Integración de la interfaz.

Tras tener acabadas las dos partes: Plugin con la funcionalidad e interfaz las uní para tener las aplicaciones finales funcionando correctamente.

10. Integración del trabajo con la empresa MNC.

Tras finalizar la aplicación y el plugin había que integrarlo en el sistema de MNC para que todo funcionara de acuerdo a su programa principal.

Puesto que la empresa ya tiene un sistema definido, para nosotros ha sido difícil acceder a él dada nuestra poca experiencia en MNC. Además, sobre todo para pasar los objetos y conectar con el servidor hemos tenido problemas que han ido solucionándose gracias al buen soporte que nos ha proporcionado Miguel Ángel Zurdo, para entonces responsable de tecnología de MNC.

SECCIÓN III: ESPECIFICACION DE REQUISITOS

1. PROPÓSITO.

El objetivo principal de esta sección es establecer y detallar todas las características que un usuario final espera de nuestro sistema.

Una vez visto los conceptos anteriores muestro ahora las verdaderas necesidades del software creado así como de los problemas y riesgos que entraña.

Esta sección está dirigida a aquellos que quieran conocer acerca del sistema y los requisitos que debe cumplir, omitiendo gran parte de los detalles técnicos que implica.

Para definir los requisitos sólo se me presentó un archivo de texto con diez páginas especificando en dos de ellas cómo debería ser mi aplicación y en otra la funcionalidad que tendría.

A medida que ha ido pasando el tiempo se han tenido varias reuniones con el director del proyecto para aprobar que los cambios realizados y las ideas del desarrollador eran buenas, pero hay que resaltar que sobre el diseño de la aplicación no se especificó nada, siendo el resultado final diseñado por el propio desarrollador desde cero al querer MNC renovar la imagen de sus aplicaciones.

2. REQUISITOS DE USUARIO.

Al asignarme el proyecto se me presentó un documento especificando los detalles acerca de lo que tenía que hacer. A medida que ha ido pasando el tiempo, con las correspondientes reuniones con el director del proyecto Patrick Sjöberg, el coordinador Juan Llorens y el responsable de tecnología M.A. Zurdo, se han ido puliendo hasta tener finalmente el proyecto resultado.

Estos requisitos implican al funcionamiento general de mi aplicación.

2.1. RELACIONADOS CON EL FUNCIONAMIENTO:

Nombre asignado	Descripción
USER1	La interfaz tiene que ser específica para funcionar en una pantalla táctil.
USER2	La aplicación tiene que ser fácil e intuitiva, para un usuario sin conocimientos de informática.
USER3	Mediante la pantalla táctil se tendrá que poder monitorizar todos los ordenadores del barco.
USER4	El sistema debe informar automáticamente de los errores o imprevistos que surjan.
USER5	La aplicación debe permitir el buen funcionamiento del PC: No ser demasiado pesada o que consuma excesivos recursos.
USER6	La aplicación está orientada al mercado naval, por tanto debe diferenciarse de los de tierra.
USER7	Debe ser un producto con funcionalidad diseñada para equipos a bordo de un barco.
USER8	El sistema debe realizar los procesos de monitorización de forma automática, sin que nadie tenga que introducir comandos.
USER9	La aplicación de monitorización debe estar diseñada para estar siempre en pantalla, incluido cuando no hay nadie trabajando con ella.
USER10	El idioma de la aplicación, no así de la memoria, debe ser el inglés porque es con el idioma que trabaja MNC.
USER11	La aplicación tiene que estar diseñada para funcionar con el paquete básico de MNC compuesto por cinco ordenadores.

Ilustración 29. Requisitos de usuario relacionados con el funcionamiento

2.2. RELACIONADOS CON LA TECNOLOGÍA

Nombre asignado	Descripción
TEC1	El marco de trabajo tiene que ser Microsoft .NET
TEC2	El lenguaje de desarrollo principal será C#.NET
TEC3	La aplicación principal se desarrollará con tecnología WPF
TEC4	El sistema tiene que contar con un servidor local instalado en el barco

Ilustración 30. Requisitos tecnológicos

2.3. RELACIONADOS CON LA PANTALLA TÁCTIL

Aquí englobo los requisitos específicos que tiene que tener la interfaz de la aplicación para pantalla táctil ya que son muy específicos.

A continuación voy a mostrar los requisitos que se me pasaron en cuanto a diseño para la GUI. Se especificó que era una sugerencia de presentación.

- La pantalla debe parecerse a la presentada en el siguiente dibujo:

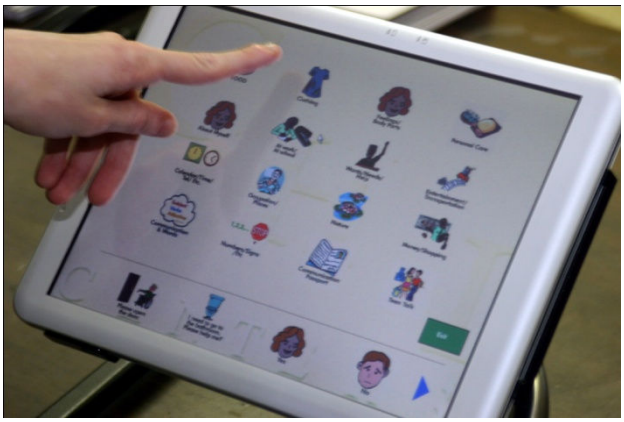


Ilustración 31. Ejemplo de aplicación para pantalla táctil

- En cuanto al modelo de pantalla táctil a utilizar no se especificó nada en concreto. Se puede usar cualquiera. El modelo que proporcionarán a los barcos está todavía por decidir.
- La pantalla táctil debe estar situado en el puente de mando.
- Se sugirió que la manera de presentar la información fuera la siguiente:

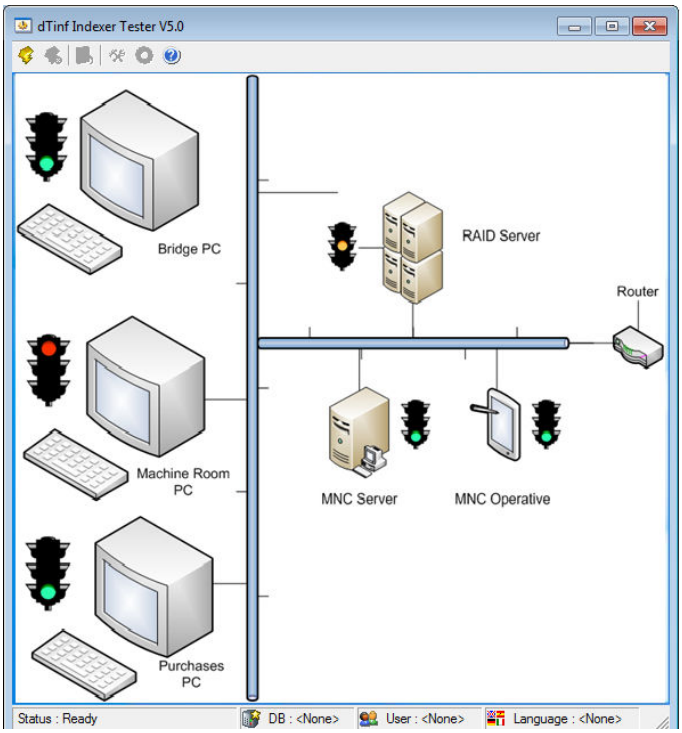


Ilustración 32. Boceto de la interfaz a implementar

- Al tocar en un ordenador, el nivel de servicios que se ofrezcan debe ser parecido al siguiente:

Donde las funcionalidades principales serían:

- Estado detallado de los ordenadores.
- Predicción de estado.
- Histórico de Backups.
- Recuperar archivos.
- Recuperar el sistema a una fecha en concreto.
- Recuperar el sistema al último estado conocido.
- Instalar otro ordenador aquí.
- Arrancar un ordenador.

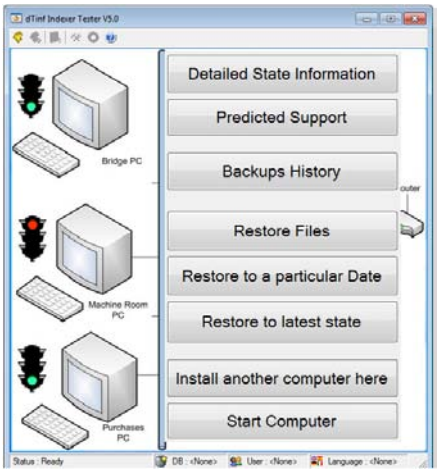


Ilustración 33. Boceto de pantalla de operaciones

Estas son las funcionalidades pensadas para mi aplicación. Mi proyecto se limitará a mostrar un estado detallado de los ordenadores, dejando el resto de las funciones para futuras implementaciones.

2.4. SOLUCIÓN A LOS REQUISITOS DE USUARIO.

USER1: La interfaz tiene que ser específica para funcionar en una pantalla táctil.

El hecho de que la interfaz tenga que ser específica para una pantalla táctil supone que debe tener unas características especiales tales como:

- Grandes espacios entre iconos para que no haya problemas al pulsar.
- Crear iconos grandes para que no haya que fijarse demasiado.
- Puede tener llamadas a funciones propias de una pantalla táctil como zoom u otras características.
- Debe tener una resolución especial que se ajuste al tamaño de la pantalla de la mejor manera posible.

Un requisito imprescindible para la consecución de este requisito es contar con una pantalla táctil de prueba, que ya ha sido solicitada al director de MNC.

USER2: La aplicación tiene que ser fácil e intuitiva, para un usuario sin conocimientos de informática.

Para solucionar este requisito es conveniente:

- Crear menús intuitivos, no utilizar abreviaturas, acrónimos o vocabulario especializado.
- Que los nombres de los iconos sean intuitivos y describan a la perfección su cometido.
- Un menú de ayuda donde se explique hasta el último detalle de lo que aparece en cada pantalla.
- Una organización muy clara. Para ello en la pantalla principal podemos ver los ordenadores, cometido principal de la aplicación. Si se quiere usar alguna función entonces iremos a un menú de funciones para gente que sabe hacerlo de tal manera que una persona que sólo está vigilando el estado de las máquinas no tenga que involucrarse en otros temas.
- Utilizar colores para expresar el estado de los ordenadores: Verde indicará que todo anda bien, amarillo que algo falla y rojo que hay un problema.

USER3: Mediante la pantalla táctil se tendrá que poder monitorizar todos los ordenadores del barco.

Al ser este el cometido principal de mi aplicación se dotará en todas las pantallas del cuadro de estado de los ordenadores, en unas se verá en un margen más pequeño y en el caso de la pantalla principal y otras en la parte central.

Para la consecución de este requisito se necesitará que MNC proporcione un entorno de trabajo con al cinco ordenadores para que estén conectados al mismo tiempo.

USER4: El sistema debe informar automáticamente de los errores o imprevistos que surjan.

Al tener siempre monitorizados los ordenadores siempre podremos percatarnos de cuando hay un problema en alguno de ellos. La aplicación está diseñada para que cuando uno de los parámetros que se controla de cada ordenador sobrepase los límites que le establezcamos se cambien determinados aspectos de la interfaz como el color de los ordenadores o el del semáforo que los acompaña.

De ese modo el usuario podrá percatarse fácilmente de que hay un problema y así intentar solucionarlo.

USER5: La aplicación debe permitir el buen funcionamiento del PC: No ser demasiado pesada o que consuma excesivos recursos.

La aplicación creada no ocupa demasiado porque no tiene mucho contenido multimedia. Al ser esto uno de los requisitos más importantes la he dotado de imágenes de pequeño tamaño que no tardan al cargar y que por tanto no consumen recursos. Ninguna de sus funciones lo hace tampoco y el programa corre sin problemas.

USER6: La aplicación está orientada al mercado naval, por tanto debe diferenciarse de los de tierra.

La aplicación se ha creado para un usuario sin conocimiento informático. Hoy en día la informática está fija en nuestras vidas nos dediquemos a lo que nos dediquemos pero las generaciones anteriores no tienen estos conocimientos. Por eso he diseñado este programa de forma intuitiva.

USER7: Debe ser un producto con funcionalidad diseñada para equipos a bordo de un barco.

La funcionalidad es una parte muy importante de la monitorización. Al tratarse de equipos que están en un barco hemos seleccionado determinados parámetros a monitorizar claves cuyo fallo supondría un problema como podría ser el porcentaje de memoria física que hay libre o si tiene conexión a internet mediante los parámetros de red.

USER8: El sistema debe realizar los procesos de monitorización de forma automática, sin que nadie tenga que introducir comandos.

Al estar el plugin instalado en todos los ordenadores de la red enviará de forma automática información al servidor cada poco tiempo. Del mismo modo mi aplicación leerá de la base de datos del servidor esta información y la actualizará, gozando al usuario de libertad absoluta. La aplicación para pantalla táctil puede estar activa durante días en una computadora encendida que siempre mantendrá actualizado el estado de los ordenadores de la red.

USER9: La aplicación de monitorización debe estar diseñada para estar siempre en pantalla, incluido cuando no hay nadie trabajando con ella.

Al ser la monitorización de ordenadores y componentes informáticos la parte más importante MNC quiere que esta aplicación esté siempre abierta mostrando al usuario el estado de las máquinas.

Para ello se ha diseñado una pantalla principal donde se puede diferenciar claramente cada uno de los ordenadores y su estado mediante colores.

USER10: El idioma de la aplicación, no así de la memoria, debe ser el inglés porque es con el idioma que trabaja MNC.

La aplicación y los comentarios han sido realizados en inglés para este cometido. En un principio para la interfaz hice traducciones al sueco y al español pero el director de proyecto no lo consideró oportuno ya que el idioma con el que MNC trabaja es siempre el inglés debido a la internacionalidad de sus clientes.

USER11: La aplicación tiene que estar diseñada para funcionar con el paquete básico de MNC compuesto por cinco ordenadores.

El proyecto reflejará los cinco ordenadores que componen el sistema, cada uno con su plugin instalado.

2.5. SOLUCIÓN A LOS REQUISITOS TECNOLÓGICOS

TEC1: El marco de trabajo será Microsoft .NET.

Se trabajará con la herramienta Microsoft Visual Studio 2008 para cumplir tal propósito y se adquirirá conocimiento en tal marco de trabajo.

TEC2: El lenguaje principal de desarrollo será C#.NET.

Usando Microsoft Visual Studio 2008 podremos compilar tal lenguaje. Previamente se formará a los desarrolladores en él.

TEC3: La aplicación principal se desarrollará con tecnología WPF.NET.

En un principio se sugirió usar Windows Forms pero finalmente este proyecto se decantó por usar WPF dentro del marco de trabajo .NET. Se especializará a los desarrolladores y diseñadores en dicha tecnología que nos proporcionará grandes ventajas.

TEC4: El sistema tiene que contar con un servidor local instalado en el barco.

La información que obtengamos de los ordenadores, se la pasaremos al servidor para que esta sea accesible desde cualquier punto de la red local directamente así como por la propia empresa desde su oficina de tierra al contactar con él.

3. REQUISITOS DE SISTEMA.

3.1. ARQUITECTURA GENERAL.

Son aquellos que implican a las máquinas sobre las que va a funcionar mi aplicación, hasta el posicionamiento en el barco o en la propia red.

MNC proporciona a sus clientes ordenadores de la marca HP (Hewlett Packard) con unas características definidas que muestro a continuación.

- Sistema operativo Windows XP.
- Ordenador completo Hewlett Packard dc7900.
    - 1 disco duro de 160 Gb.
    - Pantalla LCD marca HP, modelo L1950g de 19 pulgadas. (Resolución máxima 1280x1024 píxeles)
    - Software de ofimática: Windows XP profesional edition.
    - Periféricos incluidos: Impresora HP LaserJet P1006, ratón y teclado.
    - Software antivirus: ESET Smart Security.
    - Otro software: Microsoft Office 2007, Foxit Reader (para leer archivos con formato PDF), Cute PDF writer (Para crear archivos con formato PDF).

La arquitectura donde la aplicación estará instaurada será la siguiente:

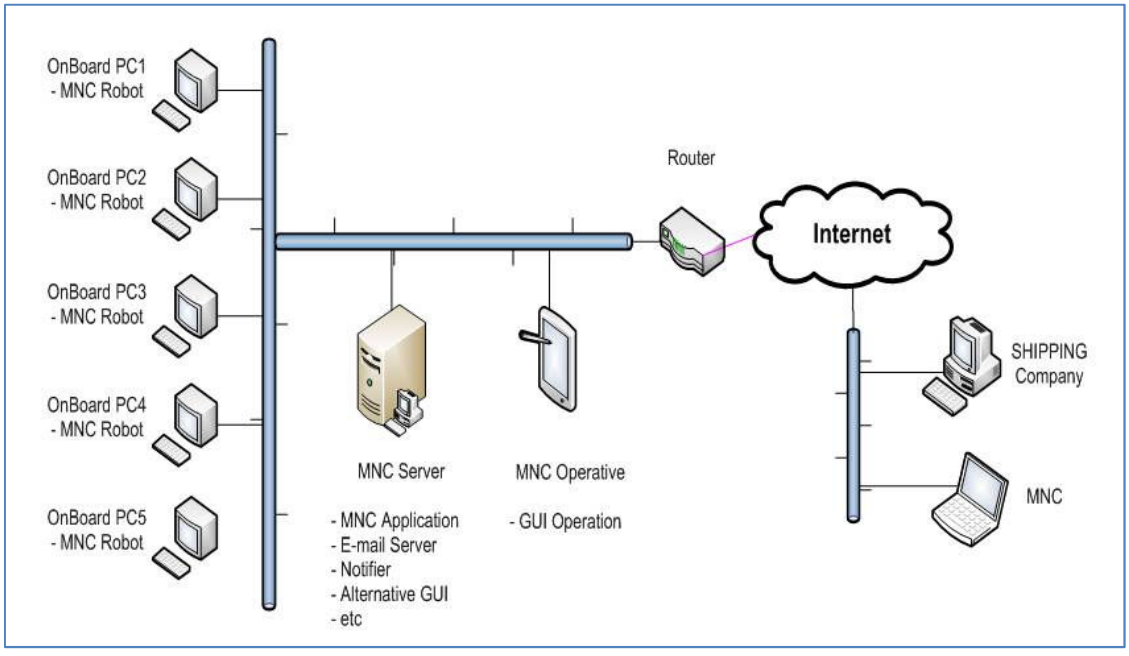


Ilustración 34. Arquitectura de sistema en un barco

Donde:

- La línea azul gruesa representa la red local que se va a crear en el barco y por tanto donde estarán conectados todos los elementos (ordenadores, servidor, pantalla táctil con la aplicación de monitorización).
- OnBoard PC1 Es el primer ordenador que vamos a monitorizar. Contiene MNC robot, que es la aplicación que contendrá la totalidad de las funciones, donde una de ella será el plugin creado en este proyecto para monitorizar su estado enviando los datos al servidor local.
- OnBoard PC2, PC3, PC4, PC5 será el resto de los ordenadores que vamos a monitorizar. Cada uno contendrá a su vez un MNC robot.
- MNC Server es el servidor local que habrá en el barco. En él se guardarán los archivos de seguridad de los ordenadores así como la base de datos que almacenara los datos del estado de cada uno.
- MNC Operative será la pantalla táctil que contenga la aplicación creada en este proyecto desde la cual se van a monitorizar todos los ordenadores para conocer siempre su estado.
- Router será el elemento que conecte todos los elementos de la red y a su vez a ésta con internet.
- Shipping company será un ordenador que esté conectado a internet y que haya contratado los servicios de MNC. De este modo podrá tener la información de lo que ocurre en su barco.
- MNC será el ordenador desde el cual se trabajará en la oficina de Mariehamn. Recibirá los datos periódicamente información de lo que ocurre en el barco a través de internet para que las oficinas de MNC para poder interactuar con ellos.

3.2. ARQUITECTURA DE PRUEBAS.

Las máquinas sobre las que se ha desarrollado y probado esta aplicación tienen las siguientes características:

Máquina 1:

- Marca y modelo: Fujitsu-Siemens Amilo 2528Xi.
- Procesador: Intel Core duo T7700 (2 núcleos) a 2.40 Ghz.
- Memoria RAM: 2 módulos DDR2 de 1 Gb. Total 2 Gb.
- Disco duro: 320 Gb.
- Sistema operativo: Windows 7 Professional edition.
- Pantalla de 17 pulgadas.
- Arquitectura de 32 bits.

Máquina 2:

- Marca y modelo: Dell Studio 15.
- Procesador: Intel Pentium i7 (4 núcleos) a 1.6 Ghz.
- Memoria RAM: 2 módulos DDR3 de 2 Gb. Total 4 Gb.
- Disco Duro: 500 Gb.
- Sistema operativo: Windows 7 Home Edition.
- Pantalla de 15.6 pulgadas.
- Arquitectura de 64 bits.

Máquina 3:

- Marca y modelo: Toshiba Satellite
- Procesador: Intel Pentium dual core a 2.0 Ghz.
- Memoria RAM: 2 módulos DDR2 de 2 Gb. Total 4 Gb.
- Disco duro: 120 Gb.
- Sistema operativo: Windows 7 Professional Edition.
- Pantalla de 15,6 pulgadas.
- Arquitectura de 32 bits.

4. ENTREGAS PARCIALES. CÓMO SE HA IDO DESARROLLANDO EL PROYECTO.

A lo largo del desarrollo del proyecto con el objetivo de la aprobación y definición de requisitos le envié al director de proyecto Patrick Sjöberg una serie de documentos explicando las ideas que tenía, lo que pensaba hacer, lo que no y su por qué. De este modo podría aclarar conceptos con ellos y definir mi aplicación a gusto de MNC.

Casi todas las cuestiones se basaron en cómo estaba organizado el sistema de MNC sobre el que iba a desarrollarse mi programa y sobre la interfaz que iba a desarrollar puesto que significa la imagen que MNC iba a dar, de modo que más que cuestiones de funcionalidad que tratamos con el responsable de tecnología Miguel Ángel Zurdo, se trataba puramente del aspecto que querían que tuviera mi aplicación.

Los originales de estos documentos fueron escritos en inglés pero han sido traducidos para poder ser leídos con facilidad en esta memoria:

4.1. PRIMERA ENTREGA (4 DE MAYO DE 2010). PRESENTACIÓN DE DISEÑO E IDEAS.

En esta primera entrega quería mostrar cómo estaba empezando a desarrollar mi proyecto a partir de la interfaz, trabajando con el aspecto de las pantallas que tendría mi aplicación y con Microsoft Expression Blend 3.0.
Creo que uno de los puntos más importantes es el diseño, la forma en que el programa se nos presenta; sobre todo en aplicaciones de pantalla táctil, así que pretendo formarme bien en este aspecto y dedicarle el tiempo necesario.
Es por eso que me decidí a usar la plataforma WPF de el marco de trabajo .NET 3.5 ayudado por la herramienta Visual Studio y Microsoft Expression Blend 3.

Con respecto a la plataforma WPF, Miguel Ángel Zurdo, la persona responsable de la tecnología en MNC, me ha hablado muy bien de esta nueva tecnología así que siguiendo su consejo voy a realizar la interfaz del programa con ella.

De modo que este primer texto es sólo para mostraros la imagen que el programa dará, el diseño, no es exactamente lo que hace el programa por sí mismo. Esta funcionalidad vendrá detallada en los siguientes entregables cuando se me dé una luz verde para continuar con esto.
Por lo tanto, pensando que el programa debería ser intuitivo y fácil porque la gente que lo use puede tener poca idea de ordenadores decidí mostrar iconos grandes y jugar con los colores.

Les muestro cómo sería la pantalla principal de mi programa.

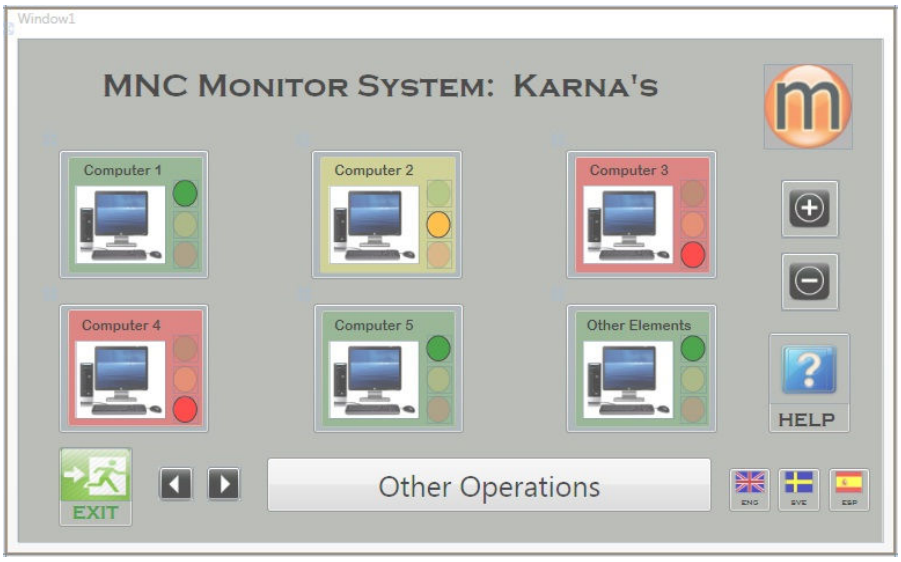


Ilustración 35. Boceto de pantalla principal de mi aplicación. (Karna's es el nombre del barco)

Estoy pensando en usar un "marco" en común para todas las pantallas o ventanas, manteniendo una serie de elementos que permanecerán siempre en pantalla independientemente de en que parte del programa estemos de modo que sea intuitivo y fácil de usar y localizar por el usuario. Las partes que quiero inamovibles son:

- Botón EXIT de salida de la aplicación.
- Iconos de navegación (flechas) por el programa.
- Iconos de selección de lenguaje.
- Botón de ayuda.
- Zoom in y Zoom out.
- Logo de la empresa MNC.

Mi idea es que los otros seis botones con cinco de los ordenadores que el paquete básico que MNC vende tiene y el icono "Otros elementos" sea movido al margen izquierdo de la pantalla una vez se salga de la principal, de ese modo uno puede seguir navegando y usando el programa pero siempre tendrá al alcance de la vista el estado de los cinco ordenadores. Imagine que estás en alta mar y uno de los ordenadores se rompe mientras operaba en otras pantallas. Creo que es de vital importancia darse cuenta, por eso pretendo siempre mostrar en pantalla el estado de todos los ordenadores.

Uno de los puntos a tener más en cuenta es que estoy basando este proyecto en una aplicación para gente no experta en el campo de la informática como capitanes de barco por ejemplo, gente que encaja con un perfil de más edad y que por tanto no está familiarizado con los ordenadores. Asumo que no tienen ningún conocimiento de informática.

Por eso en esta pantalla principal que muestro no he incluido ninguna función, sólo se trata de apariencia gráfica. Si quieres hacer algo con el programa entonces deberás pulsar el botón "Other operations" o pulsar cualquiera de los cinco ordenadores o el botón de "other elements" para ver el estado detallado de cada uno. Además se desplegará un menú con las funcionalidades que tendrá este programa en un futuro: información detallada del estado de cada ordenador (parte en la que se centra mi proyecto), back-ups y recuperación de la información…

Mi idea es no confundir al usuario con todos los componentes de la red, es un capitán de barco, no es necesario que sepa que es un servidor RAID o un Router, por eso he decidido incluir el botón "other elements", para evitar un impacto en una persona inexperta donde al lado de los ordenadores se monitorice una serie de máquinas que no sabe que son.

La explicación de cómo funcionarán los botones y sus colores vendrá a continuación, de momento pretendo mostraros las ideas de diseño que tengo:

Botones de ordenadores (1, 2, 3, 4 or 5)

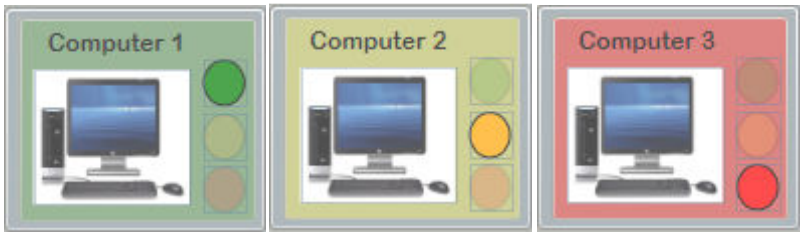


Ilustración 36. Boceto de botones de ordenador según su estado

Estos botones además del color de fondo van a tener tres luces. Dependiendo del estado de ese ordenador una de las luces va a estar encendida (transparencia opaca) y las otras dos apagadas (transparentes). Además, el propio botón que engloba tanto el ordenador como las luces será del mismo color que la luz de estado que en ese momento esté encendida, de ese modo es mucho más fácil que la persona que opera con este monitor tenga una idea de cómo funcionan las cosas más rápidamente.

Como en cualquier otra parte el color verde significa que está funcionando correctamente o el amarillo que algo funciona mal (por ejemplo, el disco duro principal tiene ocupado más del 80% de su capacidad); nada crítico, pero debe tenerse en cuenta de ahora en adelante. Por último el color rojo significa que existe un problema que debe ser solucionado ahora mismo (por ejemplo que el cable de conexión de red está desconectado o el propio ordenador apagado, que no se han hecho back-ups de la información desde hace mucho tiempo…)

Tocando uno de estos iconos irás directamente a la pantalla de estado detallado del ordenador, "detailed information" donde se mostrarán las variables que controlo de cada ordenador y el estado de cada una (nombre, localización, % de disco libre, RAM…)

Actualmente tengo una idea que podría ayudar a solucionar los problemas relacionados con la funcionalidad del programa (no sólo la que yo desarrollo sino la de toda la aplicación):

Cuando uno de los ordenadores o elementos esté en rojo o amarillo aparecerá en pantalla la información detallada. Nuestro programa sabe qué pasa, por eso estaba pensando crear un botón llamado "Soluciona el problema" o "Solve problem" que localizará donde está el problema mirando sus parámetros y sugerirá como resolverlo.

Si no es posible que el problema se arregle automáticamente entonces tendrá que mirar las sugerencias del menú de ayuda o ponerse en contacto con la oficina de MNC.

Botón de "Otros elementos" o "Other elements"

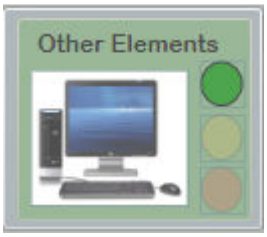


Ilustración 37. Boceto de icono de otros elementos en la pantalla principal

Tocar este botón va a hacer que se muestre un menú donde se mostrarán el resto de los elementos de la red tales como Router, servidor RAID, servidores…

Enfatizo en este punto que un capitán deberá saber qué es un ordenador pero no creo que deba saber que es un router o alguno de esos otros elementos, para esto está diseñado este botón. Aquí cada uno de estos seguirá el mismo patrón de funcionamiento y colores que los ordenadores explicados anteriormente.

Botones de bandera: Idiomas

Ilustración 38. Boceto de banderas para seleccionar el idioma de mi aplicación

La aplicación que estoy desarrollando debe ser internacional, el lenguaje no debería ser una barrera para nadie, por eso he pensado que este programa podría tener tres idiomas: Inglés (por defecto), Sueco y Español debido a los vínculos de la empresa.

Tocando una bandera se traducirá el texto automáticamente de manera permanente.

Botón de salida o Exit button

Ilustración 39. Boceto de botón de salida de la aplicación

Tocar este botón hará que se salga de la aplicación. Antes de esto se mostrará una pantalla preguntando por la confirmación de esta selección.

Botones de flechas "anterior" y "siguiente".

Ilustración 40. Boceto de botones de navegación por pantalla

Para navegar a través de las diferentes pantallas de la aplicación

Botón de ayuda, "Help button"

Ilustración 41. Boceto de botón de ayuda

El programa debe ser de fácil manejo para el capitán, por eso he pensado añadir un icono grande y azul de ayuda, en caso de que el operario esté perdido siempre podrá pulsar aquí y dependiendo de donde estés desplegar una serie de sugerencias que terminen, en caso de que el problema no se haya solucionado, contactando con la propia empresa vía telefónica.

Botones de Zoom, "Zoom in and zoom out buttons"

Ilustración 42. Boceto de botones de ampliación de pantalla

Estos dos botones llamarán a funciones de la pantalla táctil.

Logo de MNC.

Ilustración 43. Boceto de botón del logo de MNC

Este elemento tiene un gran peso en este marco que voy a usar en todas las pantallas. Siempre va a ser mostrado y mediante él se puede regresar a la pantalla principal de la aplicación. Es como un acceso directo al punto de partida.

Botón de Otras operaciones o "Other Operations button"

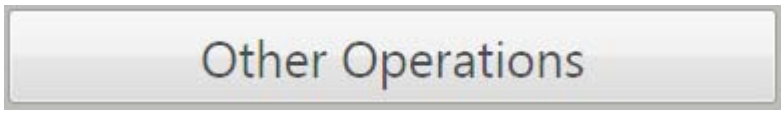


Ilustración 44. Boceto de botón para otras operaciones

Cuando quieras accede a alguna función del programa aparte de la monitorización de ordenadores debes pulsar aquí.

Después de hablar con Miguel Ángel creo que las funciones que debería tener este programa en un futuro son:

- Estado detallado de la información: Pulsando aquí aparecerá una pantalla donde elegiremos qué ordenador queremos monitorizar y al elegirlo se mostrará la misma que se mostraría cuando pulsamos sobre uno de los ordenadores o sobre el botón "otros elementos" de la pantalla principal. Será la parte principal en la que se centre mi proyecto.
- Historia de Back-ups: Tocando este botón puedes elegir un ordenador y se mostrarán los back-ups que se han realizado en un determinado periodo de tiempo (podría esto ser un filtro de búsqueda). Además creo que debería haber una opción donde se mostrara una lista con todos los ordenadores y sus respectivos back-ups con opción de impresión si es posible.
- Recuperar archivos: Llamará al programa de MNC que ya habéis desarrollado vía web.
- Restaurar a un día particular: Pulsando aquí puedes restaurar el estado del ordenador al día que tú elijas.
- Restaurar al último estado.
- Instalar un ordenador aquí: Este botón es en caso de que quieras tener los mismos archivos que tenías en otro ordenador.
- Arrancar un ordenador: Esta opción arrancará un ordenador independientemente de su localización.

De momento estas eran mis ideas, cuando despedí el proyecto pedí a mis directores y responsable que sugirieran las mejoras que vieran claras lo antes posible para poder seguir trabajando.

La respuesta por parte de Patrick y Miguel Ángel fue muy buena, a ambos les gustó mucho el diseño de la aplicación y las ideas que tenía si bien me dijeron de cambiar algunas cosas.

4.2. SEGUNDA ENTREGA (7 DE JUNIO DE 2010). PANTALLA PRINCIPAL: CÓMO FUNCIONA.

El propósito de este Segundo archivo que envié era explicar cómo iba el desarrollo de la aplicación a día 7 de junio de 2010.

Es la primera vez que hago una interfaz, así que Visual Studio es nuevo para mí así como la plataforma WPF y su lenguaje XAML, lo que requiere un tiempo de formación.

¿Cómo estoy trabajando?

Para navegar por la interfaz creo ventanas. Todo el programa funcionará con una sola abierta menos en el caso de pulsar el botón ayuda donde aparecerá otra nueva.

Al ser nuevo con estos lenguajes y plataformas estoy haciendo las cosas de una manera y quiero mostrároslas para ver si estáis de acuerdo.

Conocimientos previos que quiero inculcar antes de que se pruebe la aplicación:

- Siempre puedes regresar a la pantalla principal (Window1) pulsando el logo de MNC desde cualquier pantalla.
- Siempre vas a poder salir del programa cuando pulses el botón de cerrar ventana o cuando pulse el icono de salida.
- Botones de “Anterior”, “Siguiente” y “Zoom”: No he empezado a trabajar con ellos, lo haré cuando reciba la pantalla táctil que pedí a Patrick Sjöberg.
- Aclaraciones a los lenguajes: De momento para mostrar cómo será sólo me he limitado a traducir la pantalla principal a los diferentes idiomas. La idea es que esa traducción se aplique a todas las demás ventanas.

A continuación voy a explicar las diferentes pantallas que se muestran en mi aplicación:

WINDOW 1 Pantalla principal

Ilustración 45. Vista previa de la pantalla principal

Contiene la pantalla principal del programa donde se monitoriza. Se supone que la pantalla debe estar siempre activa estés trabajando con el ordenador o no porque este el propósito de la aplicación. Como puede verse se han realizado cambios en los colores con respecto a la versión anterior pero el patrón de trabajo es el mismo.

WINDOW 3 Menú de ayuda

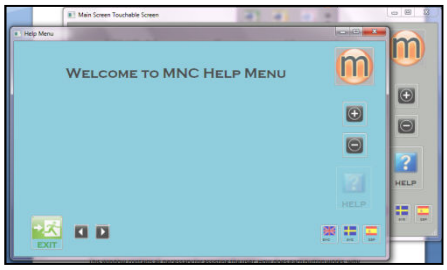


Ilustración 46. Vista previa del menú de ayuda

Esta ventana contiene todo lo necesario para ayudar al usuario. Cómo funcionan los botones de la pantalla dónde estás y que puedes conseguir pulsando en ellos.

El contenido de esta ventana variará dependiendo de la pantalla donde te encuentres. A diferencia del resto este menú de ayuda aparecerá como una nueva ventana, será la única vez que tengamos dos ventanas abiertas a la vez. Esto lo hago para que sea más intuitivo y al ver detrás la ventana principal de la aplicación uno no tenga miedo a cerrar el menú de ayuda por si cierra todo el programa.

WINDOW 4 Otras operaciones

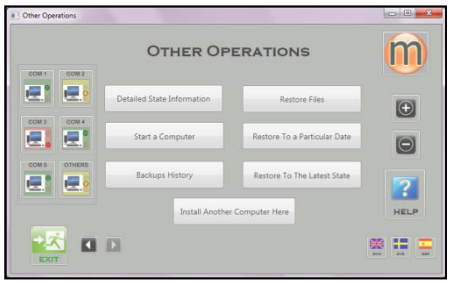


Ilustración 47. Vista previa de otras operaciones

Tal y como escribí en la primera entrega que os entregue quiero que siempre se tenga presente el estado de los ordenadores, es por eso que los incluyo en todas las pantallas en el margen izquierdo. Esta ventana contiene las funcionalidades principales que quiero que tenga mi programa.

La única función con la que he trabajado es “Detailed state information”, que permite al usuario ver el estado de un ordenador.

WINDOW 5 Selección de ordenador.

Ilustración 48. Vista previa de selección de ordenador

Esta ventana viene después de la 4 (Otras operaciones), al pulsar en "Detailed state information", muestra los ordenadores que hay en la red y los demás elementos para que tú elijas cual quieres monitorizar.

WINDOW 6 Vista de un ordenador.

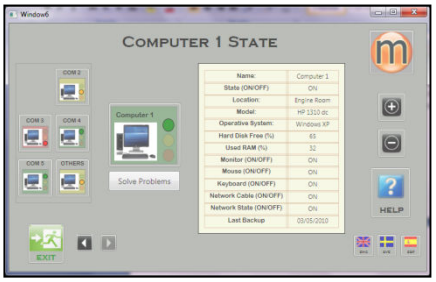


Ilustración 49. Vista previa de estado de un ordenador

Esta ventana muestra el estado del ordenador que elijo. He escrito la información que creo que se debe tener en cuenta de un ordenador pero está sujeto a vuestra opinión:

Nombre, si está encendida o apagada, donde está localizada, modelo del ordenador, sistema operativo, porcentaje de disco duro libre, si determinados periféricos están conectados o información de los backups.

No quiero incluir muchos parámetros demasiado técnicos puesto que este programa está orientado a usuarios inexpertos.

Al hablar con Miguel Ángel me ha dicho que tenéis el código para mostrar esta información y que genera un archivo XML con el estado del ordenador así que debo cargarlo y hacerlo visible para el usuario.

SECCIÓN IV: DETALLES DE IMPLEMENTACIÓN

En esta sección quiero explicar los detalles que se han tenido en cuenta a la hora de realizar la implementación de todas las etapas que han compuesto este proyecto.

Para que las explicaciones queden claras voy a mostrar de nuevo el esquema que ya mostraba en el capítulo de Background y que explicaba de un modo gráfico el funcionamiento de mi aplicación.

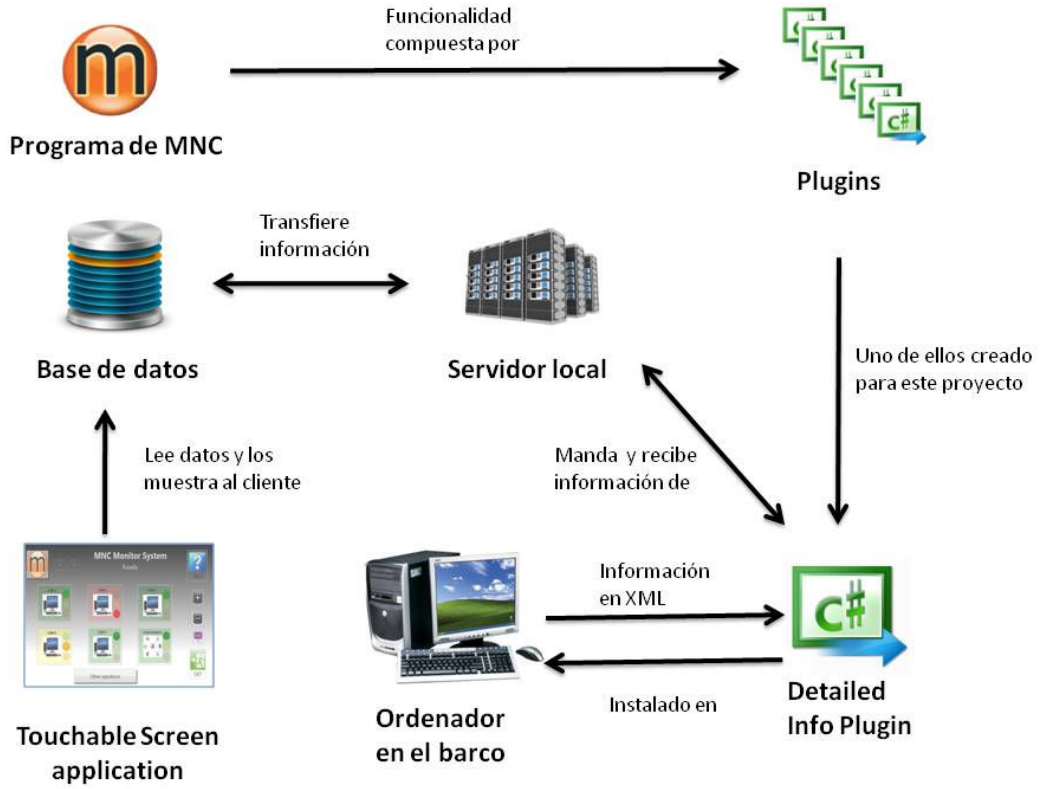


Ilustración 50. Esquema de funcionamiento general de mi aplicación

Dado que no todos los elementos han sido implementados en este proyecto quiero que el lector fije su atención en los siguientes:

- **Detailed Info plugin**. Se encarga de obtener la información de los ordenadores a monitorizar. No tiene interfaz, es sólo código escrito en lenguaje C#.NET que trabaja incluido como un plugin del programa principal de MNC "Asegure su ordenador", proporcionado a sus clientes. Al compilar el código del plugin se genera un archivo DLL que es el que se incluirá en la lista de plugins del programa principal de MNC.

Una vez el plugin detecta la información sobre el hardware del ordenador, genera un archivo XML con dicha información y lo manda al servidor del barco. Desde ahí, lo inserta en una base de datos para un mejor uso posterior. Ha sido desarrollado en lenguaje C#.NET usando Microsoft Visual Studio 2008.

- Touchable Screen Monitor Application. Constituye la parte de la aplicación que los usuarios verán y desde la que manejarán el programa. Ha sido creada usando la tecnología WPF y el lenguaje de programación C# además de XAML para el apartado gráfico.

No sólo juega un importante papel como un escenario donde presento mis objetos, sino que al haber sido creada bajo un proyecto de WPF supone el marco desde el que se llamarán a los eventos que ejecutan la funcionalidad real del programa, escrita en C# y que se encuentra en este mismo proyecto. Esta aplicación lee la información de la base de datos del servidor y a raíz de eso modifica sus parámetros para proporcionar al usuario una experiencia multimedia rica. Ha sido desarrollada con Microsoft Expression Blend 3 así como con Microsoft Visual Studio 2008.

- **La base de datos**. Se encuentra en el servidor local del barco. Para el desarrollo del programa tanto Alejandro Alonso como yo creímos conveniente la creación de una base de datos donde poder almacenar la información que vamos recogiendo para posteriormente ser leída y mostrada. Es mucha mejor opción el hecho de tener todos los datos ordenados y en un mismo sitio donde pueden ser accedidos que no en muchos archivos sueltos, solución que contemplábamos al principio.

1. DETAILED INFO PLUGIN

El plugin, que supondrá una parte de la funcionalidad del programa de MNC MobileNetControl 2.0 es una aplicación que realiza las siguientes tareas:

- Detecta los datos que le hemos indicado del ordenador donde está instalado el programa de MNC.
- Genera a raíz de ésta un archivo XML.
- Manda dicho archivo al servidor local del barco.
- Lee la información de dicho archivo y la inserta en la base de datos del servidor local.

La peculiaridad al llevar a cabo esta aplicación es que debido a requisitos de la empresa, el plugin no debe ser un programa ejecutable como tal, sino que debe ser una funcionalidad más que ofrezca el programa de MNC que queda instalado en los ordenadores a monitorizar.

Por eso, además de la fase de desarrollo como con cualquier programa, ha sido necesaria una fase adicional, la de la implantación en el sistema de MNC con su programa MobileNetCOntrol 2.0 para que trabaje como un plugin de este.

¿Por qué un plugin? De esta manera se conseguirá controlar y manipular más fácilmente la funcionalidad del programa desde MNC pues estará implantado en su sistema. En caso de que se contrate o no, podrá desactivarse al antojo de la empresa. Además de este modo podemos regular, ya que la aplicación de MNC estará siempre abierta, cada cuanto tiempo queremos que chequee el estado de los componentes del ordenador para ir actualizando la base de datos y que la información siempre sea la más actual.

1.1. ANÁLISIS

El programa MobileNetControl 2.0 es el marco sobre el que están implementada la funcionalidad en módulos llamados plugins.

Estos plugin son librerías compiladas de Microsoft Visual Studio con formato .dll. Por lo tanto, lo que teníamos que diseñar como quedó definido tras reunirnos con Miguel Ángel Zurdo es uno de esos archivos que añadiera al programa MobileNetControl la funcionalidad deseada: la detección de parámetros indicados en el ordenador donde está instalado y la inserción de estos en la base de datos a través del servidor local.

Para comenzar exploramos la carpeta donde está instalado el programa MobileNetControl. En ella existe un apartado de plugins lleno de archivos .dll. Ahí es donde teníamos que situar el nuestro, que llamamos DetailedInfoPlugin.dll.

La siguiente imagen refleja el directorio donde se encuentra el plugin (señalado) después de haber sido implementado y trasladado.

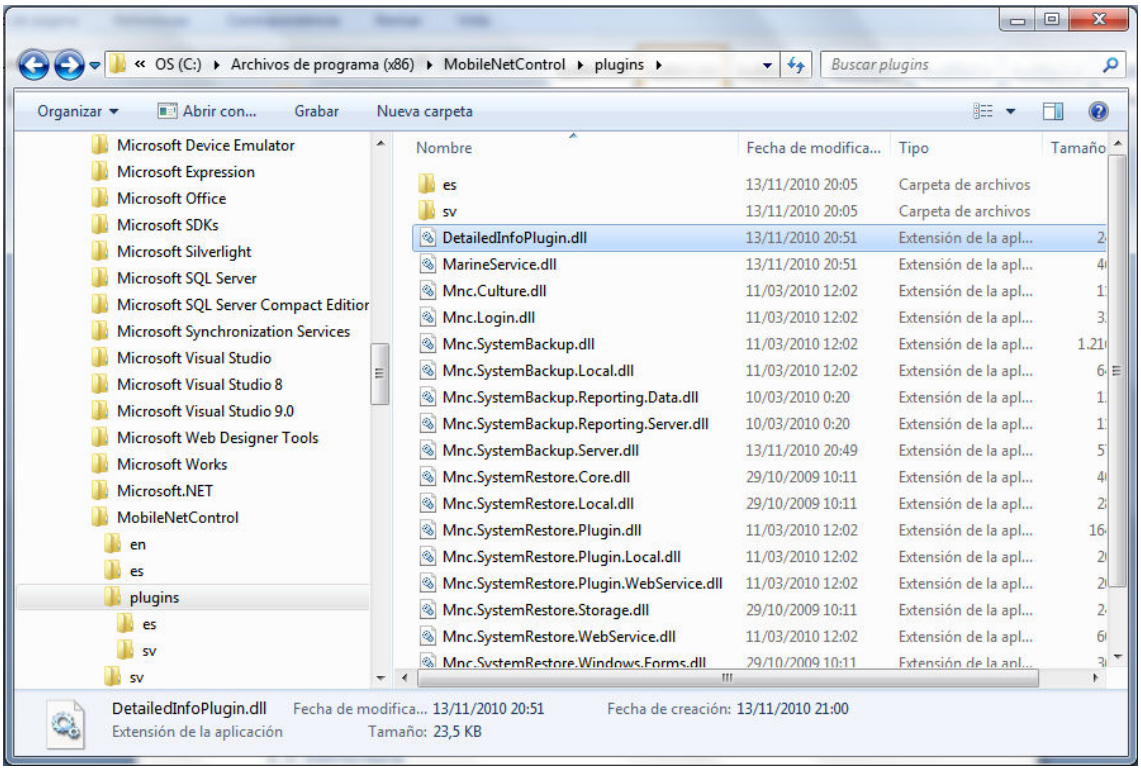


Ilustración 51. Localización física del plugin

1.2. DISEÑO

DetailedInfoPlugin es realmente una clase que hereda de la clase abstracta plugin, la cual contiene sus métodos que se ejecutarán al iniciarlo, detenerlo o simplemente mientras esta se ejecuta.

El programa estaría diseñado para funcionar en cada uno de los ordenadores de la red de uno de los barcos que ha contratado los servicios de MNC pero en nuestro caso, para desarrollarlo, lo hemos hecho sobre nuestro propio ordenador (Alejandro Alonso sobre el suyo y yo sobre el mío) porque si aquí funciona lo hará en cualquier otro ordenador donde estén instalados tanto el programa MobileNetControl como el plugin correspondiente.

El plugin se conectará al servidor local mediante una serie de parámetros que le introduciremos en el servicio que corresponde al servidor gracias al archivo “settings.xml”.

1.2.1. OBTENCIÓN DE INFORMACIÓN DEL ORDENADOR.

Para esta parte tuvimos una gran ayuda de Miguel Ángel Zurdo quien nos proporcionó un código ya implementado que, aunque tenía una funcionalidad diferente y estaba escrito en otro lenguaje del framework .NET como es VisualBasic.NET, contenía información relevante acerca de cómo hacer la operación que necesitábamos hacer.

Lo primero que debíamos hacer es tener claro qué parámetros queríamos monitorizar dentro de cada ordenador. Para ello establecimos una lista con aquellos que creíamos de más prioridad y que pueden hacer que el sistema vaya peor. El lector puede notar que la mayoría de los parámetros son estáticos, es decir, que no van a cambiar a tiempo real más que el espacio total del disco.

Esto es debido a la falta de información que ha proporcionado MNC al respecto; no han definido los parámetros a monitorizar y por tanto hemos seleccionado unos cuantos que hemos creído convenientes, siendo el único dinámico el espacio libre del disco duro, que se actualizará en función de su contenido.

La manera de la que obtenemos información del ordenador es trabajando con WMI (Windows Management Instrumentation), quien nos proporciona una serie de clases WMI que contendrán la manera de acceder a elementos hardware del equipo y así poder monitorizarlos. Esto finalmente se hace con clases Win32 de WMI, donde usaremos en concreto las relativas al disco duro, a la memoria física, el procesador, la red o la información de sistema.

Estas son sólo un ejemplo con el que se ha trabajado. Monitorizando estos parámetros quiero demostrar que el trabajo con WMI se ha cumplido y por tanto la infraestructura está creada para tal propósito. En caso de que MNC quiera cambiar los parámetros a monitorizar porque considere otros más necesarios, sólo tendrá que llamar a la clase pertinente y en concreto a la propiedad que quiera, sirviéndose de ejemplo para hacer esto del plugin que se ha creado para este proyecto. Es simplemente cambiar llamadas de funciones, existiendo un amplio abanico de posibilidades en este campo que podemos encontrar en http://msdn.microsoft.com/en-us/library/aa389273(v=vs.85).aspx.

Esta es finalmente la lista con los parámetros que monitorizamos en este proyecto:

Referente al disco duro:

- Espacio total. Con el fin de saber siempre que capacidad máxima se tiene o, en caso de que salga 0, reconocer que el disco está roto. En un principio los datos se obtienen en bytes pero con el fin de aclarar lo expresamos en Gb, que es la unidad de medida más común. Es un dato que será estático si el disco duro está funcionando correctamente.
- Espacio libre. Para saber cuanta información hay disponible en el disco duro. Es un parámetro muy importante a monitorizar ya que un sistema con poco espacio libre puede acarrear problemas. Igual que el espacio total, se expresa en Gb. Será el único parámetro dinámico con el que trabajamos, variará en casi todas las ocasiones.

Referente a la memoria RAM:

Cada ordenador va a tener dos módulos de memoria, en los que vamos a monitorizar:

- Tipo de memoria: Puede ser DDR, DDR2 o DDR3, siendo estas la penúltima la más común y la última la más novedosa.
- Velocidad a la que trabaja: Se expresará en Mhz.
- Capacidad: Cantidad de memoria que tiene. Se expresa en Gb.

A la hora de mostrar los datos no vamos a utilizar todos de los dos módulos. En el caso del tipo de memoria y la velocidad sólo utilizaremos uno porque se da por supuesto que los dos módulos son del mismo tipo.

El único caso de parámetro que se repetirá es la capacidad porque hemos creído bueno saber que cuando el parámetro es 0 significa que ese módulo está averiado.

Referente al procesador: Los datos mostrados son de uso informativo.

- Fabricante: Es información para el usuario.
- Número de núcleos.
- Modelo.
- Velocidad. Indica la velocidad de proceso de cada uno de los núcleos, puesto que en todos será igual. Este parámetro está medido sin utilizar el turbo, o sea, sin sufrir una aceleración software que tienen disponibles muchos de los modelos actuales para determinadas ocasiones. Se expresa en Mhz (Mega Hertzios).

Referente a la red:

- IP del ordenador. Contiene la dirección IP que se le asigna al equipo en concreto dentro de la red.
- Gateway. Contiene la dirección de la puerta de acceso a la red.
- Netmask. Contiene la dirección de la máscara de red.
- DNS. Contiene la primera dirección DNS. La segunda la hemos descartado por falta de espacio en la consulta.

Tras tener claro qué elementos queríamos controlar se trabajó con el plugin, se comenzó a trabajar físicamente con la clase WMI de .NET. Para cada uno de los medios en con los que queremos trabajar utilizaremos una diferente. Estas son las que hemos utilizado:

- Clase Win32_LogicalDisk para obtener la información de los parámetros del disco duro como espacio total y espacio libre.
- Clase Win32_PhysicalMemory para los datos de los parámetros de la memoria física (RAM).
- Clase Win32_Processor para los datos relacionados con el procesador.
- Clase Win32_NetworkAdapterConfiguration. Para los datos relacionados con la configuración de la red.

Además hemos utilizado otra clase para obtener información general del sistema como el usuario:

- Clase Win32_ComputerSystem para parámetros de configuración de la máquina.

Incidiendo en el dinamismo de los datos y los parámetros quiero aclarar que el hecho de tener parámetros que no varían con el tiempo tiene una explicación. Elementos hardware tan importantes como la memoria física o el procesador entre otros, pueden causar daños importantes en el equipo en caso de que se rompan o suponer grandes pérdidas de información. Por eso, al monitorizar datos como son por ejemplo la capacidad de cada módulo de memoria RAM, aunque no varíe en el tiempo puesto que es siempre la misma, queremos monitorizar el correcto funcionamiento de los módulos propios, ya que en caso de rotura mostrarán un 0 en sus valores, cambiando el estado de la interfaz WPF y alertando al usuario de que hay un fallo grave.

De este modo, aunque no tengamos detalles ni cambios en sus valores si todo funciona bien, podremos monitorizar si hay un fallo importante en elementos de tanto peso como son el procesador, la memoria RAM o el disco duro.

¿Cuándo debe obtenerse la información del ordenador?

Para tener monitorizado el sistema constantemente y que no se consuma demasiada memoria repitiendo demasiado el proyecto, hemos llegado a la conclusión de que si este plugin se ejecuta cada 12 horas (2 veces al día), se conseguirá una monitorización fiel. Este tiempo será el finalmente designado para la aplicación. Con motivo de las pruebas y presentaciones del programa, será rebajado para que pueda comprobarse su funcionamiento a tiempo real.

¿Cómo se guarda la información obtenida?

En un archivo XML, es decir, un archivo que contiene etiquetas con el nombre de cada parámetro y que se generará de forma automática y ordenada cada vez que se realice el chequeo de la máquina.

Realmente estos archivos ocupan muy poca memoria en el disco duro (no llegan al byte), puesto que sólo son archivos de texto. No sería muy problemático que la carpeta destino se llenara de estos archivos y trabajar con ellos directamente. Sin embargo, para tenerlo todo ordenado y evitar el error que puede ser trabajar con un archivo equivocado, decidimos crear una base de datos donde guardar dicha información.

¿Qué se hace con dicha información?

Se envía al servidor local del barco. Además, el plugin la lee y la inserta en una base de datos creada exclusivamente para tal propósito. De ese modo conseguimos tener toda la información de los barcos ordenada en ella para un fácil acceso y manipulación de datos.

1.2.2. GUARDADO DESTINO DE LA INFORMACIÓN OBTENIDA.

La información se guardará en una base de datos del servidor local creada con SQLite usando la herramienta SQLite administrator, como he explicado en apartados anteriores.

Su fin es tener los datos almacenados de forma ordenada para que mi interfaz acceda a ellos y así pueda mostrar al usuario el estado de todos los parámetros de la máquina donde están albergados programa, plugin y base de datos.

1.2.3. TRABAJO CON EL SERVIDOR.

Tal y como hemos indicado previamente, en el barco existen tres tipos de ordenadores diferentes:

- Los clientes, o aquellos que vamos a monitorizar.
- Aquel que contiene la aplicación para pantalla táctil.
- El servidor, que es aquel que se encargará de escribir la información del archivo XML en la base de datos.

Este servidor se llama RepositoryServiceManager. Registra una instancia RepositoryManager y realiza otras muchas operaciones que a nosotros para este proyecto no nos conciernen.

Por eso lo que hemos tenido que hacer es añadir código que proporcione al servidor la funcionalidad para poder insertar la información de los archivos XML en la base de datos.

Para conectar con el servidor nos servimos de unos parámetros que vienen especificados en el archivo instalado en el directorio del propio servicio del servidor y que se llama settings.xml.

```
<?xml version="1.0" encoding="utf-8" ?>
<RepositoryManagementServiceConfiguration>
        <port>5054</port>
        <instance>RepositoryManager</instance>
        <username>mncrobot</username>
        <password>123456</password>
        <uncpath>\\server\path</uncpath>
         <repositorydirectory>repository</repositorydirectory>
        <usersdirectory>users</usersdirectory>
        <webserverport>5077</webserverport>
        <shipname>shipname</shipname>
        <shipusername>shipuser</shipusername>
        <shippassword>shippass</shippassword>
        <notificationintervalminutes>2000</notificationintervalminutes>
        <numberComputersOnBoard>5</numberComputersOnBoard>
 </RepositoryManagementServiceConfiguration>
```

En este archivo que hemos tenido que configurar para acceder al servicio con ayuda de Miguel Ángel Zurdo se encuentra la configuración de conexión. En él podemos variar el puerto de acceso al servidor, la instancia que vamos a registrar, el nombre de usuario y contraseña necesarios para acceder al servidor (tiene que ser habilitado por un administrador, en este caso Miguel Ángel) así como el número de ordenadores que va a haber en el barco.

Este archivo es realmente importante y muchos de los problemas que se han ido dando a medida que se ha intentado integrar este proyecto con el sistema de MNC han tenido su origen en él, puesto que al haber sido creado por la empresa contiene muchos parámetros que se escapaban a nuestro entendimiento.

¿Qué flujo de datos se da en la red del barco?

No tiene sentido que la información generada gracias al plugin de nuestra aplicación sea actualizada por cada uno de los ordenadores clientes que tenemos en una red. Con 5 ordenadores podría ser viable, pero dado que esta aplicación en teoría va a ser utilizada para sistemas con más, podría ser lioso.

Por tanto, los clientes mandarán la información al servidor del barco, que será el encargado de actualizarla. De este modo conseguimos un proceso centralizado.

La pantalla táctil, por último, con la aplicación que he desarrollado, se conectará a la base de datos sólo para leer la información.

Este flujo de datos puede verse explicado gráficamente con la siguiente imagen:

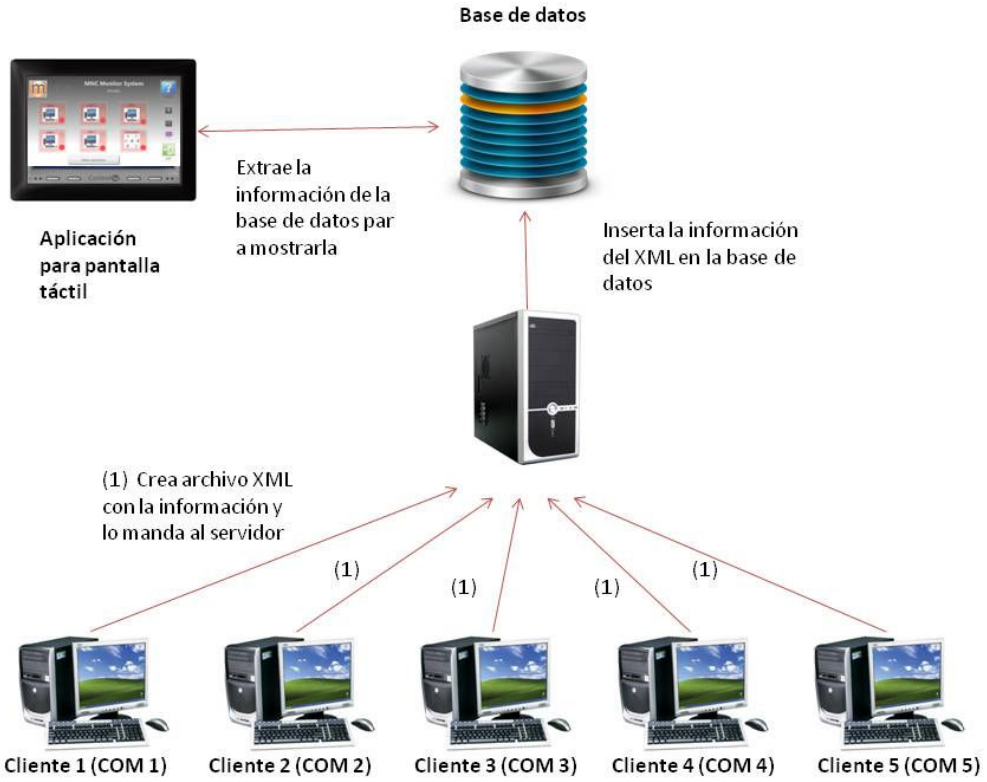


Ilustración 52. Flujo de datos general

1.3. IMPLEMENTACIÓN

En este apartado quiero mostrar sin utilizar código cuales son las clases que se van a usar en cada una de las partes que forman este sistema interconectado. En ellas quiero nombrar las librerías

Hay que tener claro que el método RUN es el que realmente empieza a ejecutar la funcionalidad del plugin, que es obtener la información del cliente. Cuando lo hace se duerme durante 12 horas hasta que vuelve a activarse para repetir la operación y de ese modo actualizar la información siempre.

1.3.1. CLIENTE

CLASE WMI

En esta clase es donde se ejecutan las consultas gracias a las clases WMI antes explicadas, para obtener toda la información del sistema.

Utiliza las siguientes librerías:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Management;
using System.Management.Instrumentation;
using System.Net;
using System.Threading;
using System.Configuration;
using System.Data.Linq;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using  Mnc.DetailedInfoPlugin;
```

Fragmento de código 23. Librerías usadas en la clase WMI

Contiene los siguientes métodos públicos:

- getComputerName. Para obtener información general del sistema como el nombre. Recordábamos que usaba la clase WMI Win32_ComputerSystem.
- getProcessorInfo. Para obtener información del procesador como el fabricante, el modelo (o familia como se llama en el código), la frecuencia de reloj máxima y el número de núcleos que contiene. Recordamos que usaba la clase WMI Win32_Processor.
- getDiskInfo. Para obtener información del disco duro como el espacio total y el espacio libre disponible. Usaba la clase WMI Win32_LogicalDisk.
- getMemoryInfo. Para obtener información de un como la capacidad, el modelo o la velocidad de un módulo de memoria RAM. Usaba la clase Win32_PhysicalMemory.
- getNetworkInfo. Para obtener información de la configuración de red.
- getMemory1Info. Para obtener la información referente al módulo de memoria 1.
- getMemory2Info. Para obtener la información referente al módulo de memoria 2.

CLASE XMLINFO

En esta clase encontramos implementada toda la funcionalidad relacionada con la creación, lectura y escritura de los archivos XML.

Utiliza las siguientes librerías:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml;
using System.IO;
using System.Xml.Linq;
using System.Data.SQLite;
using Mnc.DetailedInfoPlugin;
```

Fragmento de código 24. Librerías usadas en la clase XMLInfo

La clase XMLInfo contiene una serie de métodos públicos que están divididos según su funcionalidad. Son estos:

- getProcessorFamily y getMemoryType. Reciben el valor que se ha obtenido de la detección del modelo del procesador y del tipo de memoria respectivamente. Como esto es un número codificado, en estos procedimientos lo que hacemos es traducirlos al valor que tienen realmente mediante la sentencia case.
- returnComputerInfoInstance. Guarda en un objeto de tipo ComputerInformation la información obtenida de los métodos getXMLXXX que mencionaré a continuación.
- getXMLXXX donde XXX es el parámetro con el que nosotros queramos trabajar, como por ejemplo ComputerDate, ComputerID, ComputerName, ShipName, MaxClockSpeed, Manufacturer, FreeSpace, NetMask, etc. Estos métodos sirven para leer del archivo XML.
- setXMLXXX donde XXX es el parámetro con el que nosotros queramos trabajar como ocurría en los métodos getXMLXXX. Sirven sirven para escribir en el archivo XML.

CLASE DETAILEDINFOPLUGIN

Esta clase ejecuta toda la funcionalidad que he explicado en las clases anteriores.

Utiliza las siguientes librerías:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Threading;
using System.Xml;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Serialization.Formatters;
using Mnc.Core;
using Mnc.SystemBackup.Server;
using System.Runtime.Remoting.Channels.Tcp;
using Mnc.DetailedInfoPlugin;
```

Contiene los métodos GetRepositoryServiceManager el método Run, que tal y como he explicado antes, empieza a ejecutar toda la funcionalidad.

2. TOUCHABLE SCREEN MONITOR APPLICATION

Esta parte de la memoria se va a centrar en la aplicación creada para trabajar e interactuar con el usuario. Resolverá cuestiones sobre su funcionamiento, cómo ha sido implementada o el por qué de usar esos colores o esa estructura de modo que así el usuario podrá conocerla mejor y aprovechar todas sus posibilidades.

2.1. ¿CÓMO FUNCIONA UNA INTERFAZ EN WPF?

A la hora de crear una interfaz con WPF hay que tener claro que está bien dividida en dos partes:

- Diseño de la aplicación. Con sus contenedores, botones, formas, imágenes…
- Funciones de los botones una vez se pulsan.

La primera parte es a nivel de diseñador, la segunda es más a nivel de programador, allí es donde se escribirá la verdadera funcionalidad de la aplicación.

La interfaz de WPF funciona con ventanas que van creándose, apareciendo u ocultándose según el usuario va navegando por el programa. Al crear una ventana se crearán dos archivos con el mismo nombre pero distinta extensión para resaltar que van unidos y que tendrán este formato:

- NombrePantalla.xaml. Contendrá todo el código XAML que estará generando la parte que nosotros veremos de la interfaz.
- NombrePantalla.xaml.cs. Es un archivo compuesto por lo que ocurrirá cuando se pulsa cada uno de los botones que hemos creado en el archivo .xaml.

Por tanto, es muy importante saber que hay una gran relación entre estos archivos de modo que cada botón implementado en el primero, tiene que tener un parámetro o nombre de evento llamado Click que llevará a un método con su mismo nombre en el segundo archivo xaml.cs.

Por ejemplo, en el caso del botón de ir hacia atrás en la ventana principal que se encuentra en el gridPrevious, encontramos un botón llamado previous_bt. Este tendrá un parámetro Click = “previous_Click” que indicará que al pulsar dicho botón en nuestra interfaz se llamará automáticamente a un método contenido en el archivo .xaml.cs que ejecutará el código pertinente en C#.

Conociendo entonces el funcionamiento general de una interfaz creada en WPF voy a explicar el código escrito.

2.2. VENTANAS DE LA APLICACIÓN

La interfaz estará compuesta por una serie de ventanas que se irán abriendo o cerrando según vaya el usuario eligiendo una opción u otra.

En la elaboración de la interfaz he intentado siempre trabajar con sólo una ventana activa porque creo que es engorroso tener otras.

De cualquier modo he creído conveniente como única excepción que se abra otra ventana adicional al programa principal cuando se pulsa el botón de ayuda y cuando se pulsa en la caja de texto de la ventana de login para que aparezcan el menú de ayuda y el teclado, respectivamente. ¿Por qué? Porque no los considero parte de la funcionalidad del programa que es la monitorización, si no que son complementos que sirven para ayudar y guiar y el otro para introducir valores desde la pantalla táctil directamente en los casos que haya que identificarse en el sistema. Las ventanas que finalmente se han creado han sido las siguientes:

Número	Nombre	Descripción
1	1MainScreen	Ventana principal destinada a estar siempre activa.
3	3HelpMenuGeneral	Primer menú de ayuda especificando los controles básicos de navegación.
3.1	3.1.HelpMenuEspecific	Segundo menú de ayuda especificando las funciones de la ventana.
4	4OtherOperations	Ventana que contiene las botones para ir más allá de la monitorización.
5	5SelectAComputer	Ventana que saltará cuando sea necesario especificar uno de los ordenadores.
6	6COM1State	Ventana que monitoriza el estado de COM1
7	7COM2State	Ventana que monitoriza el estado de COM2
8	8COM3State	Ventana que monitoriza el estado de COM3
9	9COM4State	Ventana que monitoriza el estado de COM4
10	10COM5State	Ventana que monitoriza el estado de COM5
11	11OtherElementState	Ventana que monitoriza el estado del router, server, etc.
16	16Loginn	Ventana que se abre cuando sea necesario una autenticación en el sistema.
17	17TouchKeyboard	Ventana llamada desde el login que permite la introducción de datos desde la pantalla.

Tabla 7. Ventanas de mi interfaz

*La falta de los números del 12 al 15 viene dada por la supresión de dichas ventanas al final.

2.3. MODELO DE DATOS USADO PARA LAS PANTALLAS

En este apartado voy a mostrar un esquema sobre cómo he estructurado los objetos de mi aplicación en un par de pantallas, las más significativas. De este modo se pueden comprobar todas las divisiones que hay hechas con los contenedores y la forma que tiene.

Quiero resaltar que todas las pantallas de mi aplicación sin excepción tienen la misma estructura de contenedores o "Grids" que explico brevemente a continuación.

Número	Nv1	Nv2	Nv3	Nv4	Descripción breve
1	Window				*La ventana en sí.*
2		gridPrinc			*El grid que contiene todo.*
3			gridSup		*El grid de la parte superior.*
4				gridPrevious	*El grid del botón flecha atrás.*
5				gridNext	*El grid del botón fleche delante.*
6				gridLogoMNC	*El grid del botón logo de MNC.*
7			gridDer		*El grid de la parte derecha.*
8				gridHELP	*Grid del botón "HELP".*
9				gridZoomOut	*Grid del botón (-)*
10				gridZoomIn	*Grid del botón (+)*
11				gridENG	*Grid del botón de lenguajes.*
12				gridEXIT	*Grid del botón de salida.*
13			gridResto		*Grid central con el contenido.*

Tabla 8. Marco general de grids en las ventanas de mi aplicación

Esto quiere decir que la parte superior y la parte derecha (que forman el marco de todas las pantallas de mi aplicación) será el mismo de modo que en los modelos de datos que voy a incluir sólo se podrá ver la primera vez. Con el resto de los modelos voy a omitirlo una vez explicado lo anterior, centrándome sólo en los cambios que hay entre una y otra o lo que es lo mismo, en lo que ocurre en "gridResto". Gráficamente puede verse en la siguiente imagen.

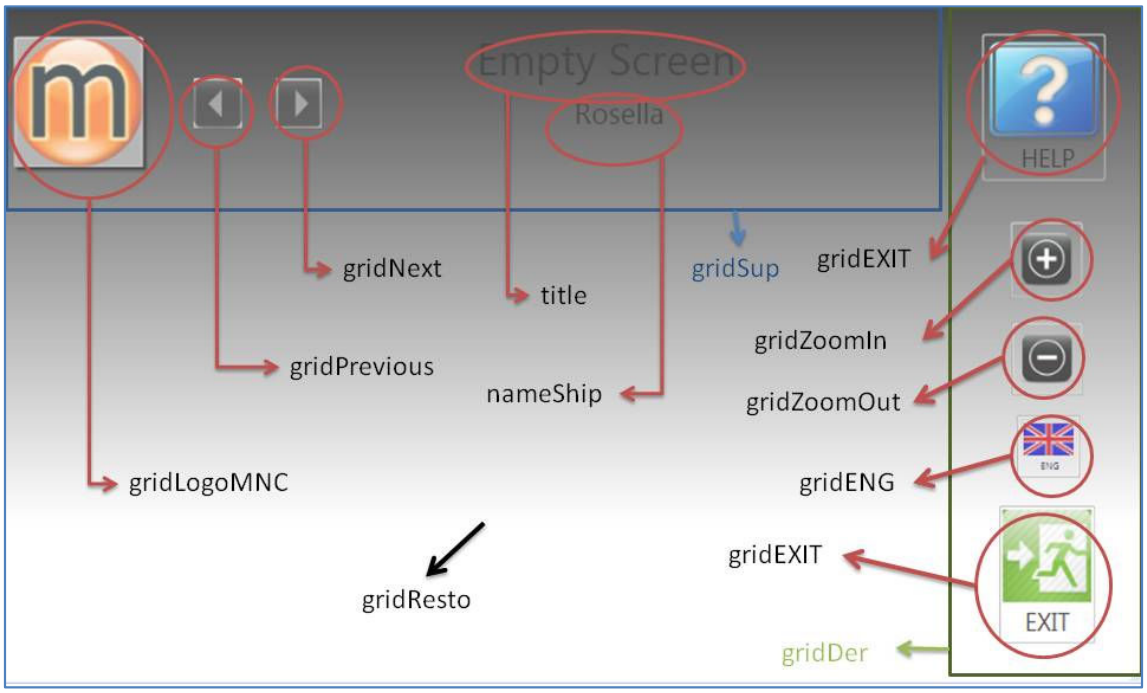


Ilustración 53. Explicación de la estructura de una ventana

De este modo los diferentes grid son los contenedores y su interior tendrá los demás objetos que se han creado.

- **Contenedores principales:** Todas las pantallas de mi aplicación tendrán 3 contenedores: gridSup, gridDer, gridResto como puede verse en la imagen.
- **Contenedores secundarios:** Para cada uno de los objetos que podemos ver en la imagen como puede ser el logo de MNC o el botón de ayuda he creado un grid que contendrá a todos los objetos que lo forman. De este además de tener la gran cantidad de objetos que se tiene ordenados bajo un mismo contenedor, es posible desplazar y aumentar o disminuir de tamaño proporcionalmente a todos los objetos moviendo simplemente el grid entre otras muchas cosas.
  En el caso del botón azul de ayuda, HELP, he creado un grid llamado gridEXIT y que contiene los siguientes objetos: La imagen azul de la interrogación, el texto donde se puede leer la palabra "HELP", un cuadrado blanco que da el efecto de la interrogación de ese mismo color y el botón que desencadena la acción deseada al pulsarlo.

2.3.1. NOTACIÓN USADA PARA LLAMAR A LOS OBJETOS.

Para llamar a los elementos he seguido un estándar donde XXX será el nombre específico de cada objeto. De modo que los nombres completos tendrán este aspecto:

- Los grid se llamaran **GridXXX**.
- Las imágenes se llamarán **XXX_im**.
- Los botones se llamarán **XXX_bt**.
- Los cuadros con texto se llamarán **XXX_textBox**.
- Los rectángulos se llamarán **XXX_rectangle**.

Y así con los demás objetos que componen la totalidad de mi aplicación. Con esto quiero ganar el fácil manejo y la universalidad en los nombres, lo que sin duda ahorra mucho tiempo a la hora de localizarlos dentro del programa.

2.3.2. PANTALLA PRINCIPAL: 1. "MAIN SCREEN".

Recordemos el aspecto que tenía esta pantalla:

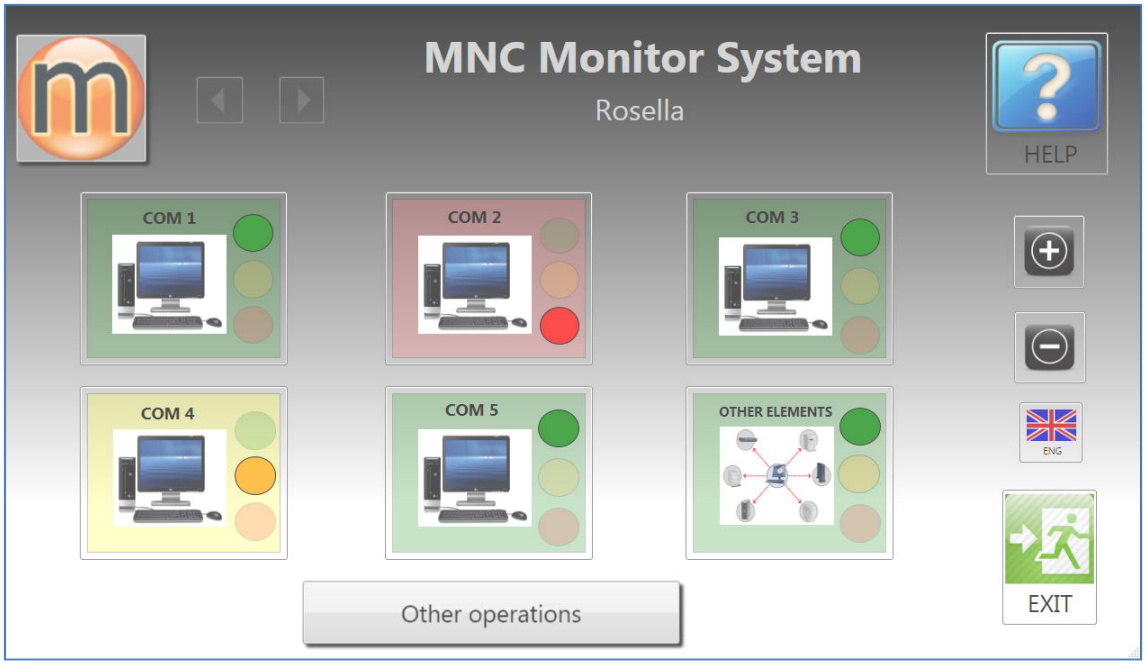


Ilustración 54. Pantalla principal de mi aplicación

Esta pantalla es la más importante ya que nos dará toda la información que necesitamos a simple vista gracias a los colores que utiliza. Por eso y porque además es parecida a la pantalla de selección de ordenador la incluyo en este apartado.

Primeros niveles y gridSup

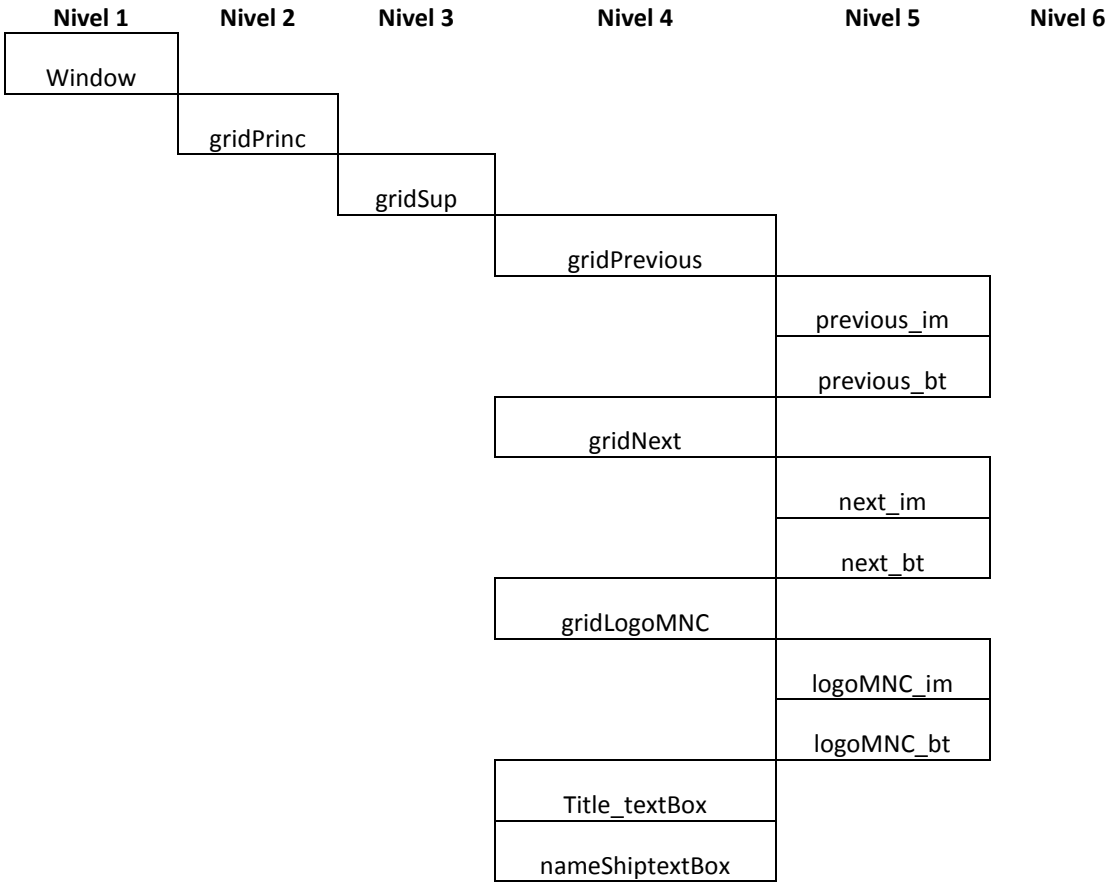


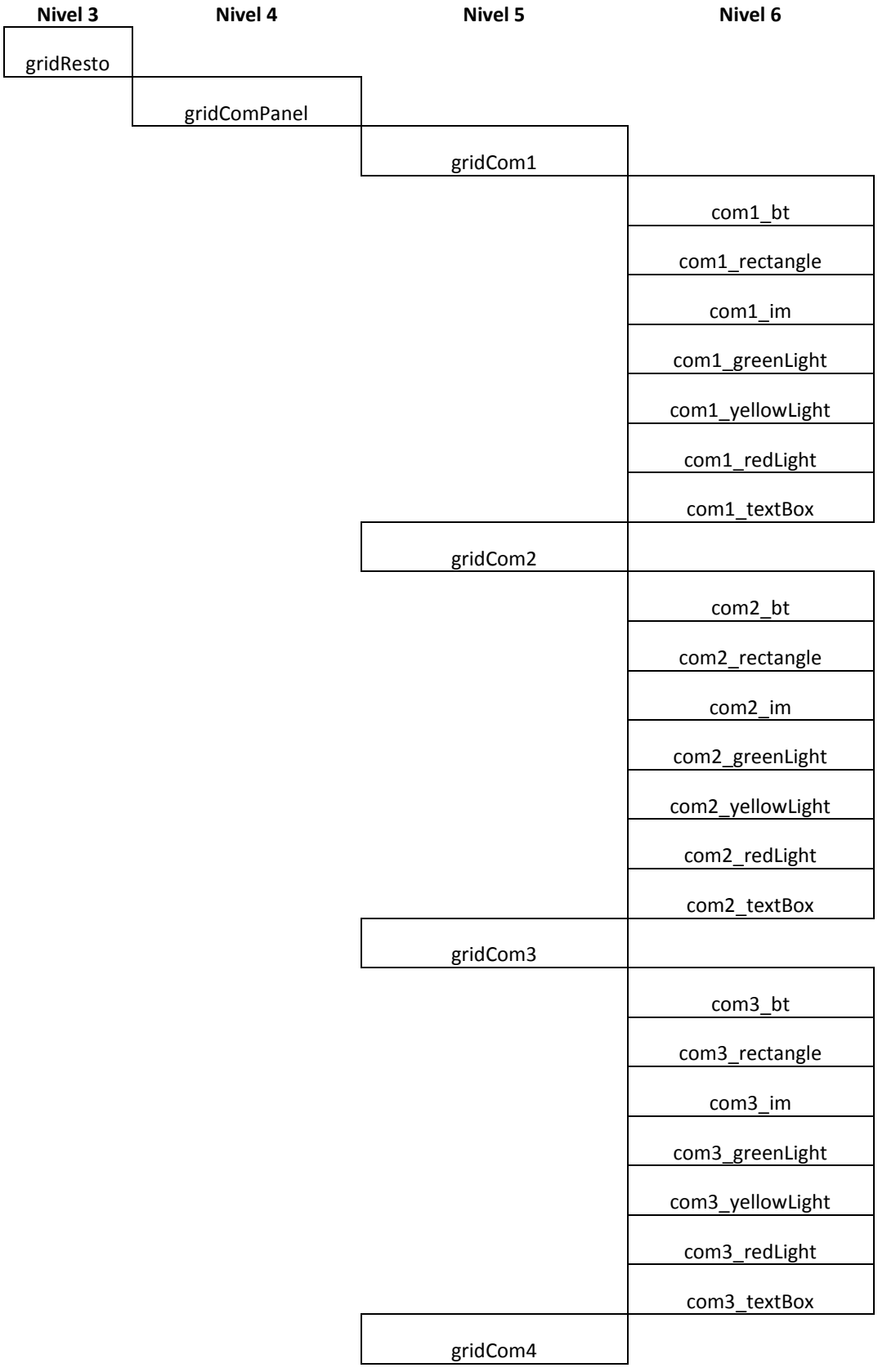
Tabla 9. Estructura detallada pantalla principal. Grid superior

Contenedor gridDer

Nivel 1	Nivel 2	Nivel 3	Nivel 4	Nivel 5	Nivel 6
		gridDer			
			gridHELP		
				HELP_whiteRectangle	
				HELP_im	
				HELP_textBox	
				HELP_bt	
			gridZoomOut		
				zoomOut_im	
				zoomOut_bt	
			gridZoomIn		
				zoomIn_im	
				zoomIn_bt	
			gridENG		
				ENG_textBox	
				ENG_im	
				ENG_bt	
			gridEXIT		
				EXIT_textBox	
				EXIT_im	
				EXIT_bt	

Tabla 10. Estructura detallada pantalla principal. Grid derecho

gridResto


Nivel 3
Nivel 4
Nivel 5
Nivel 6
gridResto
gridComPanel
gridCom1
com1_bt
com1_rectangle
com1_im
com1_greenLight
com1_yellowLight
com1_redLight
com1_textBox
gridCom2
com2_bt
com2_rectangle
com2_im
com2_greenLight
com2_yellowLight
com2_redLight
com2_textBox
gridCom3
com3_bt
com3_rectangle
com3_im
com3_greenLight
com3_yellowLight
com3_redLight
com3_textBox
gridCom4

	com4_bt
	com4_rectangle
	com4_im
	com4_greenLight
	com4_yellowLight
	com4_redLight
	com4_textBox
gridCom5	
	com5_bt
	com5_rectangle
	com5_im
	com5_greenLight
	com5_yellowLight
	com5_redLight
	com5_textBox
gridCom6	
	com6_bt
	com6_rectangle
	com6_im
	com6_greenLight
	com6_yellowLight
	com6_redLight
	com6_textBox
otherOperations_bt	

Tabla 11. Estructura detallada pantalla principal. Grid central

*Nótese que en esta última tabla omito por cuestión de espacio los niveles 1 y 2, suponiendo que son heredados de las tablas más arriba.

2.3.3. PANTALLA DE MONITORIZACIÓN: 5. COMX STATE

El aspecto que tenía esta pantalla es el siguiente:

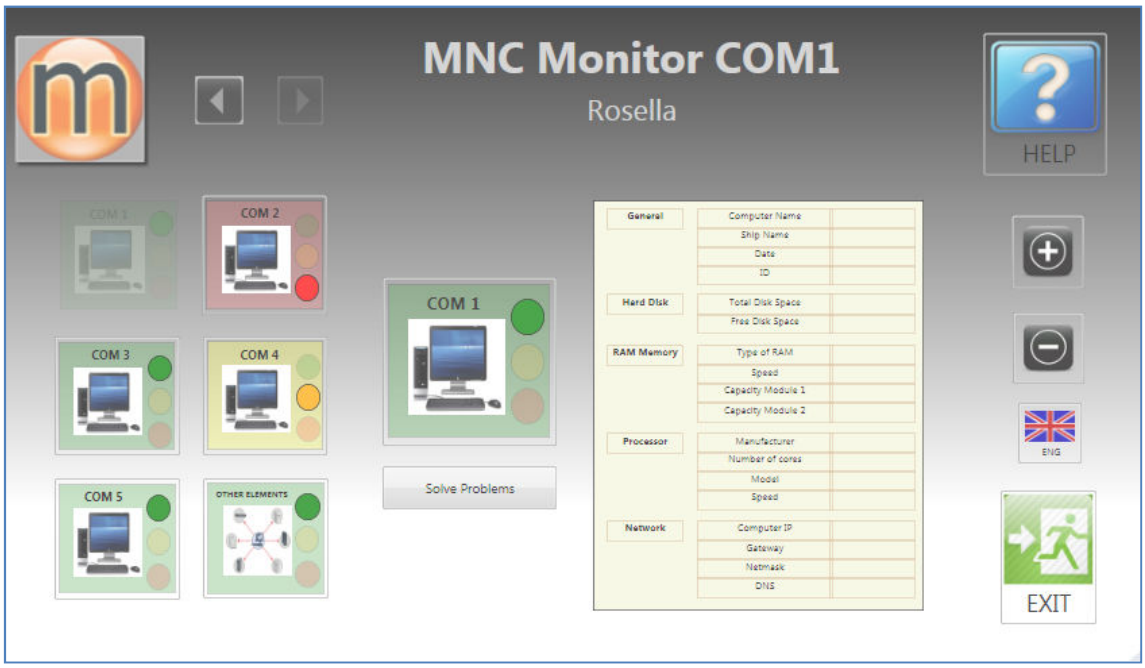
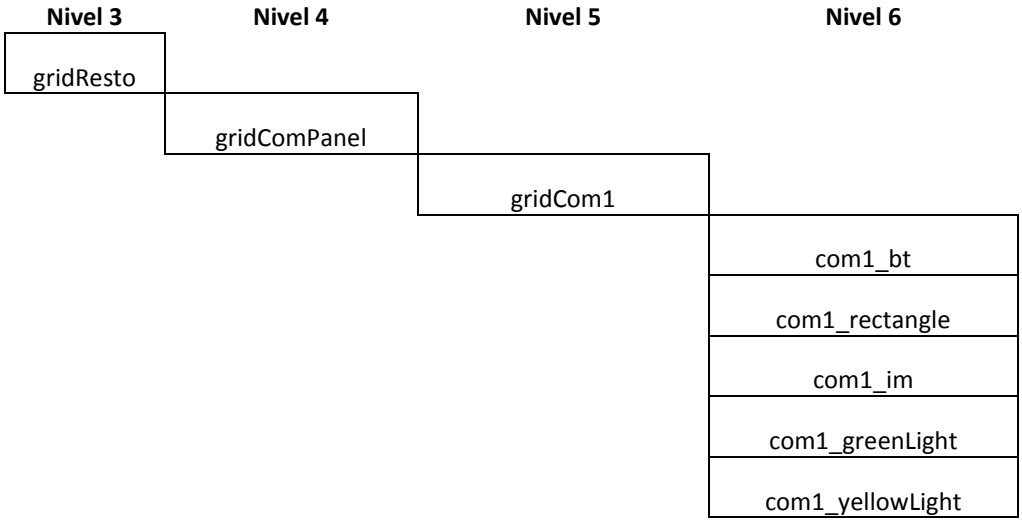


Ilustración 55. Pantalla de monitorización en mi aplicación

Esta pantalla, junto con las específicas de monitorización de los ordenadores 2,3,4 y 5 constituyen aquellas que mostrarán los parámetros del ordenador y que permitirán tanto al usuario comprobar su estado así como a la aplicación determinar qué es lo que funciona o no funciona bien.

Tal y como he indicado previamente, al ser igual el grid superior (gridSup) y el derecho (gridDer) me voy a centrar sólo en el contenido de la pantalla (gridResto).

Este es el modelo de datos con sus niveles:



	com1_redLight
	com1_textBox
gridCom2	
	com2_bt
	com2_rectangle
	com2_im
	com2_greenLight
	com2_yellowLight
	com2_redLight
	com2_textBox
gridCom3	
	com3_bt
	com3_rectangle
	com3_im
	com3_greenLight
	com3_yellowLight
	com3_redLight
	com3_textBox
gridCom4	
	com4_bt
	com4_rectangle
	com4_im
	com4_greenLight
	com4_yellowLight
	com4_redLight
	com4_textBox
gridCom5	
	com5_bt
	com5_rectangle
	com5_im

	com5_greenLight
	com5_yellowLight
	com5_redLight
	com5_textBox
gridCom6	
	com6_bt
	com6_rectangle
	com6_im
	com6_greenLight
	com6_yellowLight
	com6_redLight
	com6_textBox
gridCom1Mon	
	com1Mon_rectangle1
	com1Mon_im1
	com1Mon_greenLight1
	com1Mon_yellowLight1
	com1Mon_redLight1
	com1Mon_textBox1
	com1M_bt
gridDataPanel	
	rectangle2
	general_textBox
	name_textBox
	name_textBox1
	ship_textBox
	ship_textBox1
	date_textBox
	date_textBox1
	ID_textBox

ID_textBox1
hardDisk_textBox
diskSpace_textBox
diskSpace_textBox1
freeSpace_textBox
freeSpace_textBox1
RAM_textBox
typeRAM_textBox
typeRAM_textBox1
speedRAM_textBox
speedRAM_textBox1
capacity1_textBox
capacity1_textBox1
capacity2_textBox
capacity2_textBox1
processor_textBox
manufacturer_textBox
manufacturer_textBox1
cores_textBox
cores_textBox1
model_textBox
model_textBox1
procSpeed_textBox
procSpeed_textBox1
network_textBox
IP_textBox
IP_textBox1
gateway_textBox
gateway_textBox1
netMask_textBox

	netMask_textBox1
	DNS_textBox
	DNS_textBox1
solveProblems_bt	

Tabla 12. Estructura detallada pantalla monitorización. Grid central

2.3.4. PANTALLA DE OPERACIONES: 4. "OTHER OPERATIONS"

Por ser esta ventana diferente a los demás he decidido incluirla en este apartado.

Nótese que en este caso, igual que en los anteriores, se encuentran monitorizados todos los ordenadores, pero el hecho de haber creado contenedores (grid) para cada uno de ellos es posible cambiarlos de tamaño cuando queramos sin problema.

Recordamos como era la pantalla "Other operations"

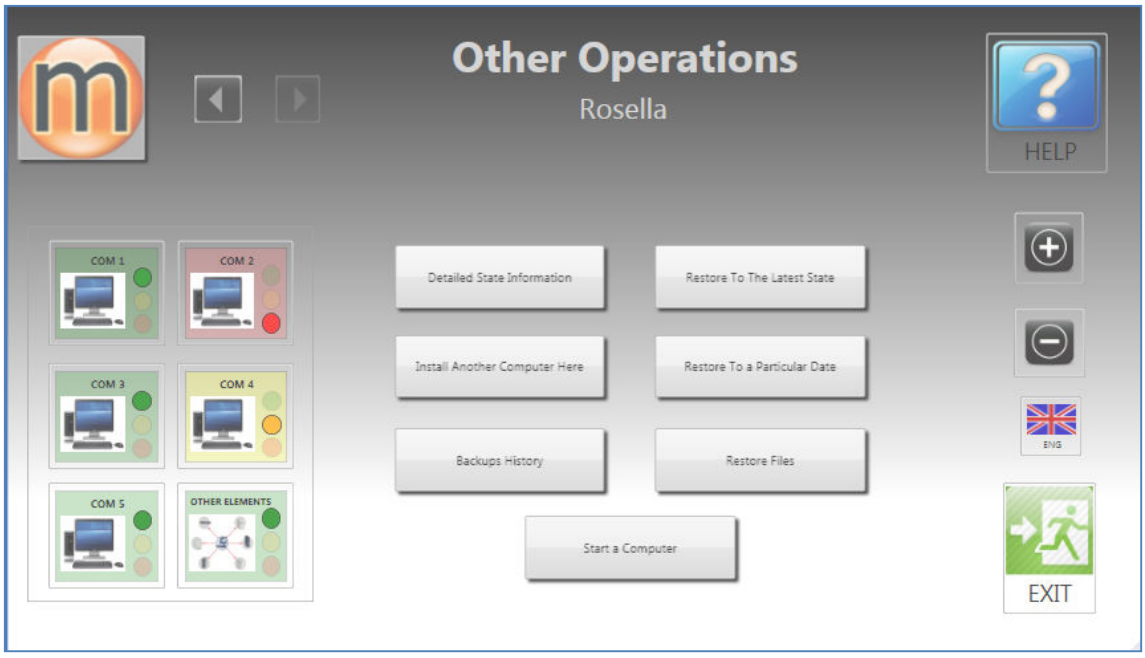
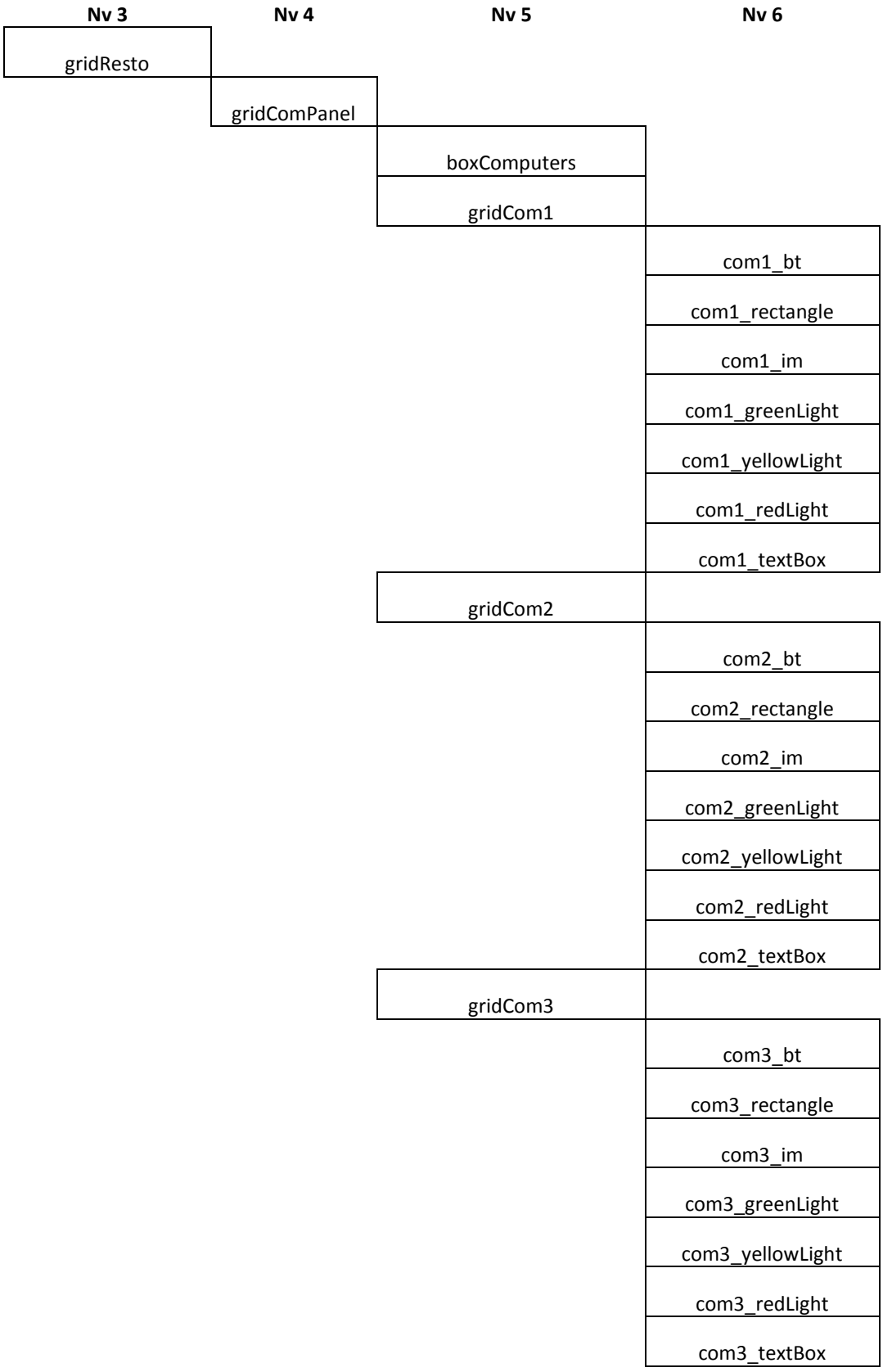
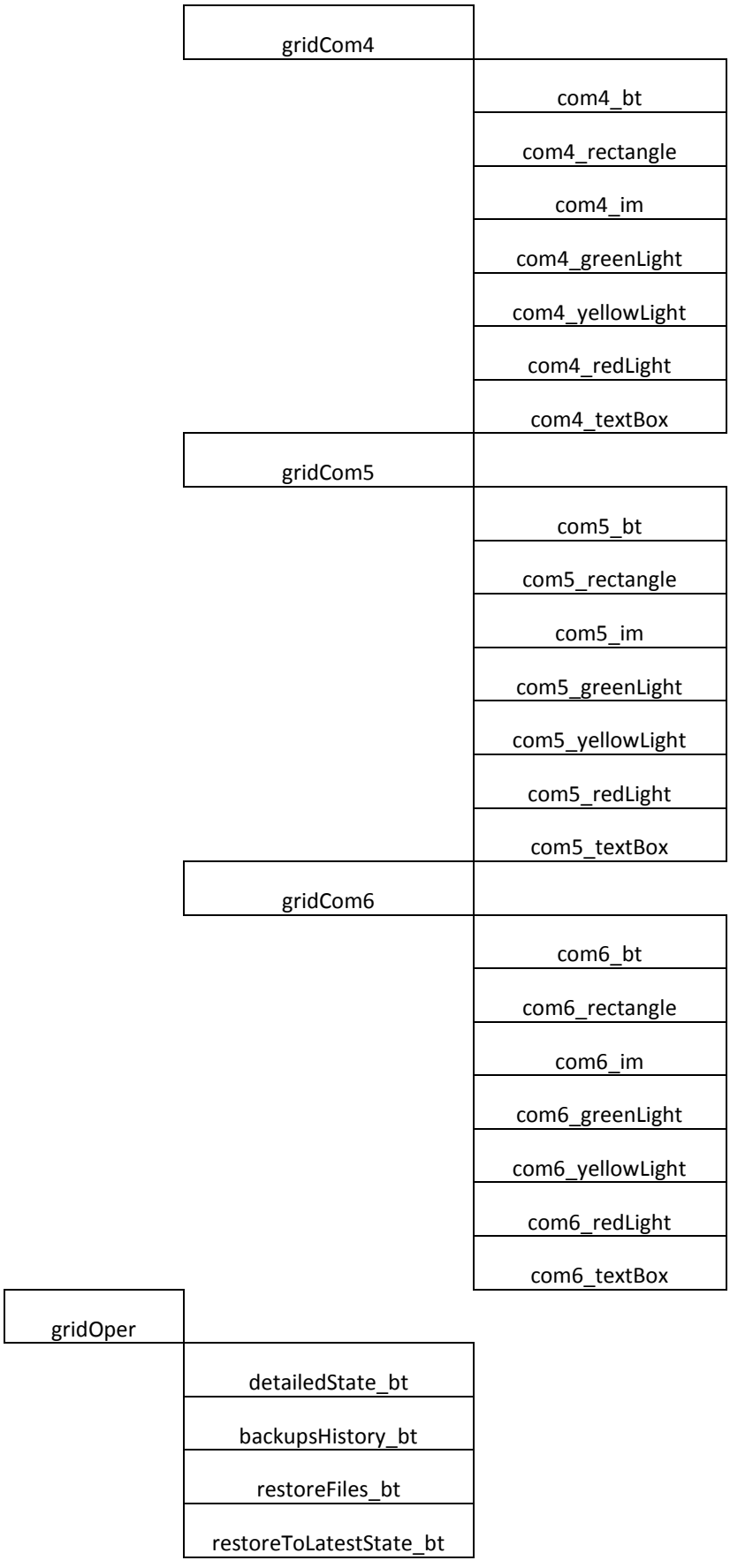


Ilustración 56. Pantalla de otras operaciones en mi aplicación

Donde el modelo de datos del grid central sería el siguiente:




gridCom4
com4_bt
com4_rectangle
com4_im
com4_greenLight
com4_yellowLight
com4_redLight
com4_textBox
gridCom5
com5_bt
com5_rectangle
com5_im
com5_greenLight
com5_yellowLight
com5_redLight
com5_textBox
gridCom6
com6_bt
com6_rectangle
com6_im
com6_greenLight
com6_yellowLight
com6_redLight
com6_textBox
gridOper
detailedState_bt
backupsHistory_bt
restoreFiles_bt
restoreToLatestState_bt

installComputer_bt
startComputer_bt

Tabla 13. Estructura detallada pantalla otras operaciones. Grid central

2.4. LISTA DE DEFECTOS

1. **Pantalla Keyboard:** Al pulsar en las teclas no se refleja en la caja de la parte superior. Como esto no se cumple, tampoco se pasa a la ventana anterior donde tendría que estar para que el usuario pudiera identificarse con éxito.

2. **Teclas de navegación por la pantalla:** Muchas veces las teclas estarán desactivadas y otras, sobre todo al retroceder, llevarán al usuario hacia la pantalla principal. Esto último está hecho para intentar solventar este problema y que finalmente el usuario tenga un punto de referencia en la pantalla principal.

3. **Teclas de zoom in y out:** Estos dos botones con el signo (+) para el zoom in y (–) para el zoom out estarán desactivados siempre debido a la falta de una pantalla táctil para efectuar las pruebas.

4. **Pantallas de ayuda:** En este caso la pantalla de ayuda general será la misma para todas las pantallas pues enseñan cómo manejarse por el marco único que todas contienen. Sin embargo, la pantalla de ayuda específica sólo ha sido implementada para “Main Screen” o pantalla principal. El resto de menús para el resto de las pantallas deberá ser implementado en un futuro.

5. **Funcionalidades no incluidas en esta aplicación:** Aunque mi proyecto sólo tenga que monitorizar el estado de los ordenadores he creído conveniente incluir en la interfaz una serie de botones que llevarán al programa en un futuro a llevar a cabo dichas funciones tales como “Restaurar el ordenador al último estado”, “Restaurar el ordenador a una fecha dada”, “Arrancar un ordenador”, etc. Por tanto, al intentar pinchar en ellas va a abrir por defecto la ventana de login simulando que es una acción que requiere autenticación en la base de datos de MNC.

2.5. SOFTWARE REQUERIDO

Hay que diferenciar dos niveles:

- Ejecución de la aplicación.
  Para ejecutar la aplicación no se hacen falta muchos programas, sólo tener un sistema operativo Windows y tener instalado el Framework .NET que por defecto viene incluido en los sistemas más nuevos.
  Si se quiere que la aplicación realice su trabajo correctamente habrá que tener abierto el programa de MNC que obtendrá mediante el plugin la información de los ordenadores.
- Modificación de la aplicación.
  Para esto hace falta tener instalado Visual Studio. Además si se quiere una experiencia de trabajo mejor puede ayudarse uno de Microsoft Studio Expression Blend 3.

2.6. ESTRUCTURA

En esta sección quiero explicar de manera gráfica cómo puede uno navegar entre las ventanas de mi aplicación pulsando los diferentes botones.

Cada una de las cajas del esquema que represento a continuación representa un modelo de pantalla. Contiene una imagen de cómo es en realidad, su nombre y un número en la esquina inferior derecha que indica el número que tiene asignado esa ventana dentro del programa, para aquellas personas que una vez lo tengan abierto o estén programando para él, puedan aclararse mejor.

El texto que aparece con cada flecha indica el botón que tiene el usuario que pulsar para saltar de una ventana a otra.

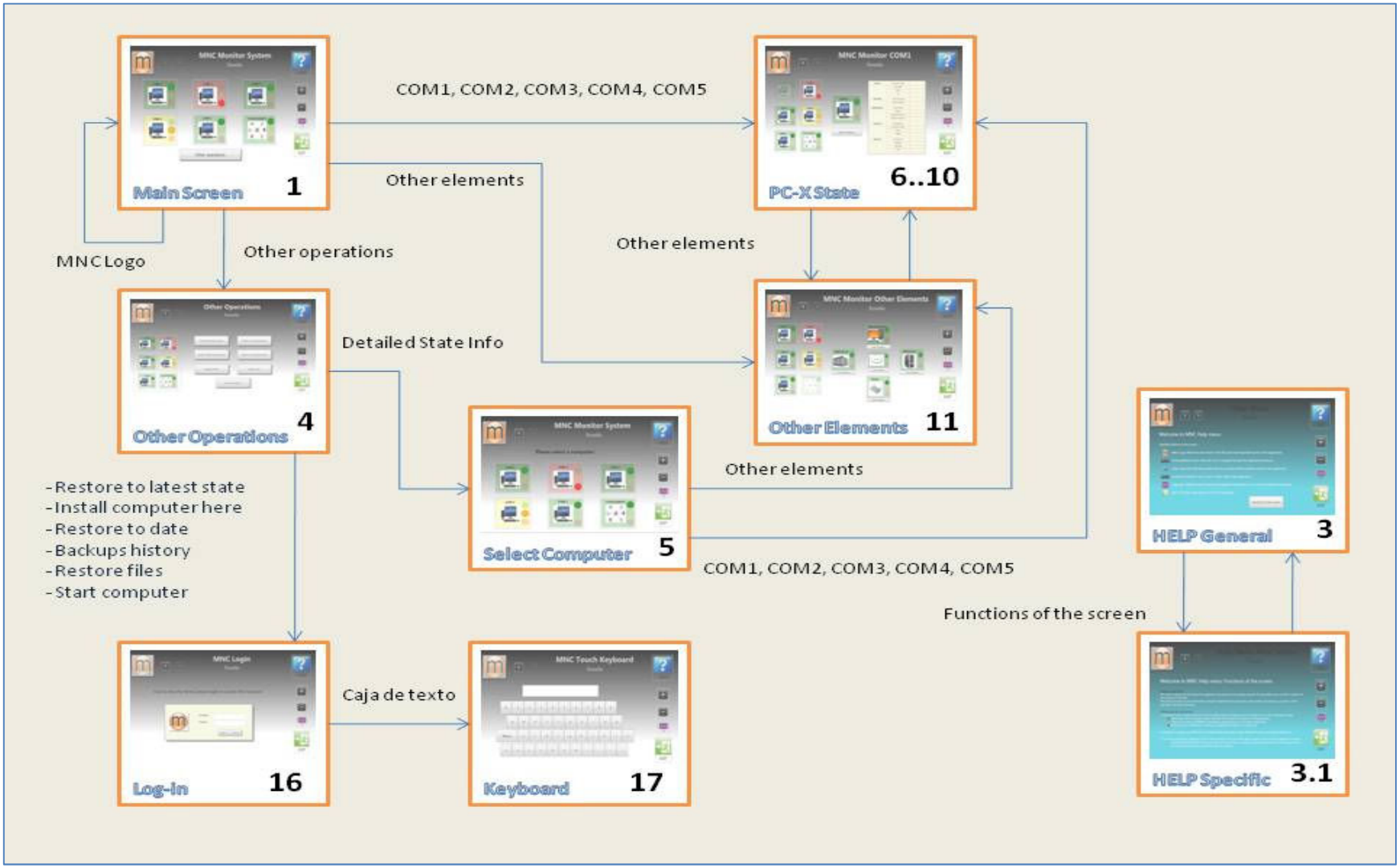


Ilustración 57. Diagrama de navegación por ventanas de mi aplicación

- Desde todas las pantallas se puede regresar a la pantalla principal “Main Screen” o (1) pinchando sobre el logo de MNC.
- Desde todas las pantallas se puede salir pinchando sobre la imagen verde de salida o pulsando el icono de salir en la ventana.
- Las ventanas de ayuda (3 y 3.1) son las únicas que se abren por separado sin cerrar la aplicación principal.
- Al pulsar cualquier botón de “Solve Problem” o al acceder a una función no disponible hasta la fecha se saltará a la venta de Log-in (16).

2.7. EL CÓDIGO DE LA APLICACIÓN.

Tal y como he comentado antes, para hacer una aplicación en WPF se usarán dos códigos, una para la parte de diseño que se hará en XAML y otra para la funcionalidad de los botones, que se escribirá en C#.NET.

2.7.1. PRIMERA PARTE: EL CÓDIGO XAML.

La interfaz está construida gracias a las herramientas Visual Studio 2008 y a Microsoft Studio Expression Blend 3, quienes facilitan al usuario la creación de una interfaz gráfica de manera esquemática generando el código de la aplicación por debajo.

De ese modo se crea la interfaz gráfica; uno tiene una "pizarra" delante a la que arrastrará y dará formato a todos los elementos que quiere añadir tales como botones, rectángulos, grids… mientras que por detrás la propia aplicación está generando el código respectivo en XAML. Así se evita tener que estar programando en un lenguaje que más que complicado es pesado de escribir puesto que tiene mucha sintaxis y muchos parámetros. Sin embargo y aunque no queramos, hay ocasiones en las que los programas mencionados no te dejan modificar una serie de parámetros o simplemente no te dan opción, por eso es necesario tener conocimiento del código, para poder escribirlo o modificarlo al antojo de cada uno directamente.

Por eso tanto Visual Studio como Expression Blend ofrecen la posibilidad de ver el programa de tres maneras. Cada una es buena dependiendo del usuario y de la tarea que esté realizando. Estas son:

- Ver sólo la "pizarra" donde arrastraremos los elementos. De esta manera podríamos crear el programa sin necesidad de tocar código XAML directamente. Es una buena vista para crear el diseño de la interfaz.
- Sólo el código XAML donde veríamos las líneas que compiladas quedarían con el aspecto de la denominada pizarra. Sirve para modificar aspectos así como estructurar los objetos que hemos creado.
- Modo Split, un modo con las pantalla dividida en dos partes donde en una podemos ver la pizarra y en la otra el código. Esta es la más completa para repasar el trabajo realizado con la interfaz.

Como ejemplo voy a escribir el código de alguno de los elementos que más se han repetido en esta interfaz como pueden ser un **grid** denominado gridPrevious que recordamos, es el grid que contiene todos los elementos referentes al botón de retroceso en la navegación del programa dentro del grid superior. Dentro de él podemos encontrar **una imagen** llamada previous_im y **un botón** llamado previous_bt.

```
<Grid x:Name="gridPrevious" Margin="208.875,70.11,800.792,47.98" HorizontalAlignment="Left"
VerticalAlignment="Top" Opacity="0.3">

        <Image x:Name="previous_im" Height="53.406" Stretch="Fill"
        Source="Resources\tr_blackII.jpg" Opacity="0.4" ClipToBounds="False"
        SnapsToDevicePixels="True" IsEnabled="True" Grid.IsSharedSizeScope="False"
        AllowDrop="True" RenderTransformOrigin="0,5" Margin="2.5,2.167,2.833,0"
        HorizontalAlignment="Left" VerticalAlignment="Top">

        </Image>

        <Button x:Name="previous_bt" Background="Transparent" FontFamily="Copperplate Gothic
        Bold" FontSize="16" Foreground="Transparent" Height="53.406" Opacity="1"
        ClipToBounds="False" SnapsToDevicePixels="True" IsEnabled="True"
        Grid.IsSharedSizeScope="False" AllowDrop="True" RenderTransformOrigin="0,5"
        Click="previous_Click" Margin="2.5,2.167,2.833,0" VerticalAlignment="Top"
        d:LayoutOverrides="Width, HorizontalMargin" IsHitTestVisible="False">

        </Button>

</Grid>
```

Fragmento de código 25. Ejemplo de un grid, el gridPrevious de mi aplicación

En el código mostrado podemos ver entonces la sintaxis en XAML de:

- Un grid: “gridPrevious”.
- Un botón: “previous_bt”.
- Una imagen: “previous_im”.

En azul he querido resaltar donde se comienza a escribir hasta que se acaba cada uno de los objetos mencionados.

En rojo he querido resaltar cada uno de los parámetros que contiene cada elemento y en verde el nombre de la propiedad o evento que se referirá al archivo con el código en C#.

Como puede observarse sobre todo en el caso del botón, hay una gran cantidad de ellos, por eso es trabajoso escribir código en XAML y la mayoría de los programadores o diseñadores prefieren utilizar programas como Expression Blend o Visual Studio.

En negro simplemente he representado los valores que tendrán los parámetros de esos tres elementos.

2.7.2. SEGUNDA PARTE: EL CÓDIGO C#

Realmente éste es el código que realiza todo el trabajo y modifica los elementos de la interfaz definidos en la parte anterior.

Además de contener el código que se ejecuta cuando se pulsa un botón, contiene el resto de la funcionalidad del programa.

Hay que tener en cuenta que el chequeo del estado de los ordenadores se hace cada vez que se abre una ventana, de modo que toda la funcionalidad se encontrará en el archivo .cs de la ventana que se abre en cuestión, no dentro del botón que abre la propia ventana.

Cada uno de esos archivos .cs que tiene cada ventana contiene una cabecera donde se llamará a los namespace necesarios para ejecutar la aplicación, los que en mi caso son:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data.SQLite;
using Mnc.DetailedInfoPlugin;
```

Fragmento de código 26. Namespace que se usan en mi aplicación

Después se tiene que definir el propio namespace donde vamos a trabajar para dicha ventana, o sea, donde entrará todo el código C# de ese archivo. Una de las condiciones es que deberá tener el mismo nombre del proyecto. Una vez definidos los namespace que usaremos, la aplicación tendrá el siguiente aspecto:

```
namespace WpfApplication2
{
   public partial class Window1 : Window
   {
      public Window1()
      {
         InitializeComponent();
         checkState1();
         checkState2();
         checkState3();
         checkState4();
         checkState5();
                  checkState6();
            }
   }
   private void Window_Loaded(object sender, RoutedEventArgs e)
   {
   }
}
```

Fragmento de código 27. Formato de código C# en WPF

Dentro podemos observar la línea `public partial class Window1 : Window` que indica la clase ventana con la que vamos a trabajar y las operaciones que vamos a hacer con ella, en este caso inicializarla.

En esta misma función Window1 es donde incluimos llamadas a las primeras funciones o funciones en sí que queremos que se ejecuten en primer lugar al crear la instancia de ventana. Por tanto, en mi proyecto, es aquí donde estarán las funciones de chequeo de estado de ordenadores, donde dependiendo de la ventana lo pintarán en pantalla y también donde se modificarán los elementos de la interfaz de un modo u otro.

El por qué de haber colocado estas funciones aquí es que la información que se vea siempre sea la correcta, evitando de ese modo que alguna vez se muestre información obsoleta.

Siguiendo con las funciones de chequeo de ordenadores, el código que ejecutan es el siguiente:

```
public void checkState1()
    {
        List<ComputerInformation> infoList = ComputerInfoList();
        ComputerInformation info = infoList.ElementAt(0);
        string name = info.ComputerName;
        string ship = info.ShipName;
        string date = info.Date;
        string id = info.ComputerId.ToString();
        string totalSpaceS = getSpaceInGb(info.TotalSpace);
        string freeSpaceS = getSpaceInGb(info.FreeSpace);
        string RAMType = info.MemoryType1;
        string RAMspeed = getMhz(info.Speed1);
        string capacity1 = getSpaceInGb(info.Capacity1);
        string capacity2 = getSpaceInGb(info.Capacity2);
        string manufacturer = info.Manufacturer;
        string cores = info.NumberCores;
        string model = info.Family;
        string procSpeed = getMhz(info.MaxClockSpeed);
        string ip = info.Ip;
        string gateway = info.Gateway;
        string netMask = info.NetMask;
        string dns = info.Dns;
            //Luego hace operaciones de condición para cambiar las luces de colores, pero al ser sentencias
            IF triviales sólo voy a incluir una a modo de ejemplo que evalúa que el espacio en el disco duro
            esté entre el 10 y el 20%, tras lo cual cambiará el marco del ordenador y las luces a amarillo.
        long totalSpace = Convert.ToInt32(totalSpaceS);
        long freeSpace = Convert.ToInt32(freeSpaceS);
```

```
        //Colours of the Computer rectangle and DISK SPACE
                    if ((freeSpace <= 0.2 * totalSpace) & (freeSpace > 0.1 *   totalSpace))
        {
          this.com1_rectangle.Fill = Brushes.Yellow;
          this.com1_redLight.Opacity = 0.2;
          this.com1_greenLight.Opacity = 0.2;
          this.com1_yellowLight.Opacity = 1.0;
              }
          }
```

Fragmento de código 28. Función general de comprobación de estado de un ordenador

La información que obtienen y con la que trabajan se ha sacado de otros procedimientos que voy a mostrar a continuación.

```
internal List<ComputerInformation> ComputerInfoList()
{
    string query = buildSelectQuery();
    string connStr = "DataSource=..\\..\\BBDD\\LOCAL_INFO.s3db;FailIfMissing=false;";
          System.Data.SQLite.SQLiteConnection cnn = new SQLiteConnection(connStr);
    System.Data.SQLite.SQLiteCommand cmd = cnn.CreateCommand();
    cnn.Open();
    cmd.CommandText = query;
    List<ComputerInformation> list = new List<ComputerInformation>();
    System.Data.Common.DbDataReader reader;
    try
        {
          reader = cmd.ExecuteReader();
          {
            reader.Read();
            ComputerInformation ci= new ComputerInformation();
```

```
            ci.Date = reader.GetString(0);
            ci.ComputerId = reader.GetInt16(1);
            ci.ComputerName=reader.GetString(2);
            ci.ShipName=reader.GetString(3);
            ci.TotalSpace=reader.GetString(4);
            ci.FreeSpace=reader.GetString(5);
            ci.Capacity1=reader.GetString(6);
            ci.MemoryType1=reader.GetString(7);
            ci.Speed1=reader.GetString(8);
            ci.Capacity2=reader.GetString(9);
            ci.MemoryType2=reader.GetString(10);
            ci.Speed2=reader.GetString(11);
            ci.Manufacturer=reader.GetString(12);
            ci.Family=reader.GetString(13);
            ci.NumberCores=reader.GetString(14);
            ci.MaxClockSpeed=reader.GetString(15);
            ci.Ip=reader.GetString(16);
            ci.Gateway=reader.GetString(17);
            ci.NetMask=reader.GetString(18);
            ci.Dns=reader.GetString(19);
            list.Add(ci);
                        }
        }

}
```

Fragmento de código 29. Función específica de obtención de los datos del ordenador

Aquí se está conectando con la base de datos y guardando la información en otra clase incluida en el archivo "ComputerInformation.cs" de la que luego extraemos los resultados a las variables de nuestro programa para poder trabajar con ellas libremente.

Realmente la consulta que hace nuestro programa a la base de datos es la siguiente:

```
string query=@"SELECT * FROM COMPUTER_INFO ORDER BY DATE ASC;";
```

Fragmento de código 30. Consulta SQL que hago en mi aplicación en C#

Después de hacer las comprobaciones pertinentes para todos los ordenadores y componentes se pasa a los métodos en los que escribimos qué se hace cuando se pulsa cada botón. Aquellos que sólo se dedican a abrir y cerrar ventanas tienen este aspecto:

```
//MNC orange logo button. Returns the user to the main screen.
        private void logoMNC_Click(object sender, RoutedEventArgs e)
        {
            Window1 wind1 = new Window1();
            this.Close();
        }
```

Fragmento de código 31. Código de evento del botón del logo MNC

Donde lo que se hace aquí al pulsar el botón logoMNC es regresar al usuario a la pantalla principal creando una instancia del objeto Window1 al que llamaremos wind1, y cerrando la actual en la que estemos para que no haya dos ventanas abiertas a la vez.

Este código y más, se incluye en todas las ventanas en las que hay que mostrar el estado de los ordenadores, bien sea en grande, o en el margen izquierdo. Esto puede parecer pesado pero es la manera de trabajar en WPF y así nos cercioramos de que siempre se ejecute el código y la pantalla siempre muestre los últimos datos obtenidos de la base de datos.

3. LA BASE DE DATOS

Para poder almacenar correctamente la información acerca de los ordenadores que mi aplicación monitoriza decidimos Alejandro Alonso y yo crear una base de datos aunque esto supusiera tener que formarnos en SQL.

Para hacerlo necesitábamos una herramienta administradora de bases de datos, por lo que elegimos SQLite administrator, gratuito, completo y que no requiere instalación.

La base de datos juega un papel importante en este proyecto almacenando los datos de los ordenadores que se monitoriza. Se realizan básicamente dos operaciones con ella que muestro en el siguiente esquema:

Ilustración 58. Interactuación con la base de datos

Por tanto creamos una base de datos llamada LOCAL_INFO que contuviera una tabla llamada COMPUTER_INFO y que estuviera formada por:

- Columnas que representan atributos que se medirían del ordenador.
- Filas que representan el ordenador que se está monitorizando.

Nombre de la base de datos: LOCAL_INFO

Nombre de la tabla: COMPUTER_INFO

Campos:

Nombre	Tipo de datos	Restricciones	Descripción
date	TEXT	NOT_NULL, SEC_KEY	Fecha de la inserción de datos
computer_id	INTEGER	NOT_NULL, KEY	ID que MNC asigna a cada ordenador que maneja
computer_name	TEXT		Nombre del ordenador
ship_name	TEXT		Nombre del barco donde está el ordenador
total_space	TEXT		Espacio total del disco duro
free_space	TEXT		Espacio libre del disco duro
capacity1	TEXT		Capacidad de memoria RAM del primer módulo
memory_type1	TEXT		Tipo de RAM del primer módulo
speed1	TEXT		Velocidad de RAM del primer módulo
capacity2	TEXT		Capacidad de memoria RAM del segundo módulo
memory_type2	TEXT		Tipo de RAM del segundo módulo
speed2	TEXT		Velocidad de RAM del segundo módulo
manufacturer	TEXT		Fabricante del procesador
family	TEXT		Familia del modelo de procesador
number_cores	TEXT		Número de núcleos
max_clock_speed	TEXT		Velocidad de reloj
ip	TEXT		Dirección IP asignada al ordenador
gateway	TEXT		Puerta de acceso del ordenador
netmask	TEXT		Máscara de Red del ordenador
dns	TEXT		Dirección DNS del ordenador

Tabla 14. Campos de la base de datos

Hay que tener en cuenta que esta base de datos ha sido creada en común entre Alejandro Alonso y Vicente García para trabajar ambos con ella desde sus respectivos proyectos.

Por eso y por temas de ordenación hay algunos datos que mi aplicación no necesita y que son los siguientes:

- Memory_type2: Considero que no es necesario mostrarlo ya que el tipo (DDR2, DDR3…) será el mismo que el del primer módulo de memoria RAM, por eso unifico ambos tipos y lo nombro en mi proyecto: Type of RAM.
- Speed2: Por el mismo motivo que con el tipo de memoria, considero que ambas memorias trabajan a la misma velocidad. Sin embargo con Capacity1 y 2 si trabajo porque si hay algún error estos valores se volverán 0.
- MAC: Este es un atributo que necesita Alejandro Alonso para mandar datos a través de internet y que yo no necesito.

SECCIÓN V: PLANIFICACIÓN DEL PROYECTO.

En esta sección del proyecto voy a explicar paso a paso lo que se ha ido haciendo desde que comencé este proyecto hasta que lo acabé. Estos resultados se mostrarán de forma escrita ayudados por archivos adjuntos que incluyo en esta memoria y que traducen el esfuerzo en dos unidades, coste y duración.

Antes de empezar quiero aclarar que una serie de condiciones referentes a todos las personas que incluyo y que han trabajado activamente en este proyecto:

- La jornada laboral diaria dura 8 horas.

- Se trabaja un total de 20 días al mes.

- A las personas se les pagará por día trabajado dado que no se puede pagar a este tipo de profesionales por horas al tener que involucrarse mucho en el proyecto que llevan a cabo.

1. PERSONAS IMPLICADAS EN EL PROYECTO.

1.1. ROLES DEL PROYECTO.

Quiero asignar tres roles que asignaré a las personas que han participado activamente en este proyecto. Los valores de los salarios vienen libres de impuestos puesto que ya han sido pagados (vendrían a ser un 40% en Finlandia).

- Ingeniero sénior: Una persona con experiencia y habilidades que se encargará de la planificación del proyecto así como de la organización. Se le asignará un salario de 116 euros al día* (2320 euros al mes contando los 20 días).

- Ingeniero junior: Una persona generalmente joven y sin experiencia en este tipo de trabajos. Se encargará de la programación y de las tareas menores que el ingeniero sénior del proyecto le asigne. Le asignaré un salario de 70 euros al día* (1400 euros al mes).

- Diseñador: Una persona sin conocimiento de programación. Dado que este proyecto trabaja con pantallas táctiles hay que contratar a uno porque el diseño es una parte muy importante. Pasará el diseño a los ingenieros para que estos lo desarrollen. Gracias a WPF podrá hacer un diseño práctico para los desarrolladores de la funcionalidad. Se le asignará un salario de 73 euros al día* (1460 euros al mes).

*Para calcular los salarios he buscado la media anual en una base de datos de salarios http://www.salarytrack.co.uk para la profesión determinada en Estocolmo (una de las ciudades más cercanas a Mariehamn).
Al darse la cifra en libras esterlinas he convertido el resultado a euros (a día de hoy el factor de conversión es: 1 libra = 1.16 euros.
Al resultado obtenido le he restado el 40% de tasas aproximadamente que tiene una persona que trabaja en las islas Åland donde se ha desarrollado el proyecto y lo he dividido entre 12 pagas para obtener el salario mensual.

1.2. PERSONAS QUE HAN PARTICIPADO EN ESTE PROYECTO.

Vicente García Adánez. Autor del proyecto y de esta memoria. De ahora en adelante V. García.

Alejandro Alonso Herrera. Autor de un proyecto paralelo al mío para la misma empresa. Al tener una gran parte en común, se le puede considerar colaborador directo del proyecto y de esta memoria. De ahora en adelante A. Alonso.

Miguel Ángel Zurdo*. Responsable de tecnología de MNC. Ha sido la persona que ha tomado las decisiones técnicas y prestado soporte en la realización de este proyecto. De ahora en adelante M.A. Zurdo.

Juan Llorens Morillo*. Coordinador del proyecto asignado. Consiguió ponernos en contacto con MNC para que éste diera comienzo. De ahora en adelante J. Llorens.

Patrick Sjöberg*. Director del proyecto asignado. Una vez Juan Llorens me puso en contacto con MNC Patrick se encargó de explicarnos la estructura de la empresa así como de qué buscaban ellos d este proyecto. De ahora en adelante P. Sjöberg.

*Estas personas trabajan activamente con MNC y por tanto no han sido incluidas en el presupuesto de este proyecto puesto que han participado indirectamente aportando información.

1.3. ASIGNACIÓN DE PERSONAS POR ROL.

- Diseñador: Vicente García Adánez.
- Ingeniero sénior: Vicente García Adánez.
- Ingeniero junior: Vicente García Adánez y Alejandro Alonso Herrera.

2. TAREAS PRINCIPALES DEL PROYECTO.

MNC necesita que se desarrolle un software para una pantalla táctil desde la que se pueda monitorizar. Esta decisión se la comunica Patrick Sjöberg a Juan Llorens, quien contacta conmigo para llevar a cabo este proyecto.

Por tanto, estas tareas que incluyo y que considero ligadas a mi proyecto son posteriores a la descripción del programa que MNC quería y que me proporcionó Juan Llorens.

La persona encargada ha sido la que ha llevado a cabo la totalidad de esta mientras que las personas colaboradoras son personas que han participado indirectamente aportando información.

1. Recopilación de información referente a la funcionalidad del proyecto.

Concierto de reuniones con diferentes personas para poder tener una especificación de requisitos del programa requerido por MNC.

Persona encargada de esta tarea: V. García, A. Alonso.

Personas colaboradora: J.Llorens, P.Sjöberg.

2. Planificación y coordinación de tareas en el proyecto.

Persona encargada: V. García.

3. Documentación e información de la plataforma .NET y C#.

Obtener información de la plataforma a utilizar y de las posibilidades que ofrece.

Persona encargada de esta tarea: V. García.

Personas colaboradoras: A. Alonso, M.A. Zurdo.

4. Documentación acerca del diseño de interfaces con WPF.

Obtener información de la plataforma WPF así como de diseño de interfaces orientadas a pantallas táctiles.

Persona encargada de esta tarea: V. García.

5. Diseño y desarrollo de la interfaz de la aplicación.

Persona encargada: V. García.

Personas colaboradoras: M.A. Zurdo, A. Alonso.

6. Desarrollo de la funcionalidad de la aplicación.

Personas encargadas: V. García, A. Alonso.

Personas colaboradoras: M.A. Zurdo.

7. Integración de interfaz y funcionalidad.

Persona encargada: V. García, A. Alonso

8. Pruebas.

Persona encargada: V. García, A. Alonso.

9. Documentación

Persona encargada: V. García, A. Alonso.

3. PRESUPUESTO ESTIMADO DEL PROYECTO.

Esta sección contiene el presupuesto que se estima del coste del proyecto durante los 6 meses de duración de la beca en Finlandia. Fue realizado en Mariehamn justo después de haberse asignado el proyecto actual.

3.1. PRESUPUESTO DE COSTES ASOCIADOS A LA VIDA EN ÅLAND.

Presupuesto de costes puntuales

				Costes	
Dia	**Mes**	**Concepto**	**Duración (h)**	**Concepto**	**Precio**
3	Enero	Reuniones skype	1	Internet mes	0,03
11	Enero	Viaje Madrid-Mariehamn	7	Viaje de ida a Mariehamn	165
1	Febrero	Reunión en oficina de MNC	2	Especificar requisitos	-
12-14	Febrero	Viaje a Helsinki	72	Conocimiento general	50
16	Febrero	Reunión con el director del proyecto	1	Avances del proyecto	-
1	Marzo	Recogida de pantalla táctil	0,5	Pruebas del programa	-

2	Marzo	Reunión con el director del proyecto	1	Presentación de ideas interfaz	-
12-14	Marzo	Viaje a Estocolmo	72	Conocimiento general	50
16	Marzo	Reunión con el director del proyecto	1	Rediseño de requisitos	-
30	Marzo	Reunión con el director del proyecto	1	Presentación de la interfaz de la aplicación	-
8-10	Abril	Viaje a Tallin	72	Obligatorio para curso universitario	100
13	Abril	Reunión con el director del proyecto	1	Avances del proyecto	-
15	Abril	Viaje a Madrid	144	Viaje a Madrid por motivos personales	180
19	Abril	Reunión en Leganés	2	Introducir tecnologías y definir requisitos	-
20	Abril	Llegada a Mariehamn	0		-
27	Abril	Reunión con el director del proyecto	1	Especificación de requisitos funcionalidad	-
4	Mayo	Presentación aplicación	2	Muestra de la funcionalidad del programa	-
11	Mayo	Reunión con el director del proyecto	1	Avances del proyecto	-
25	Mayo	Reunión con el director del proyecto	1	Avances del proyecto	-
8	Junio	Presentación aplicación	1,5	Muestra de la funcionalidad del programa	-
15	Junio	Reunión Skype coordinador	1	Planificación del proyecto	0,03
22	Junio	Reunión con el director del proyecto	2	Completar documentación	-
22	Junio	Devolver pantalla táctil	0,2		-
28	Junio	Viaje Mariehamn-Madrid	9	Vuelta del periodo de estancia en Mariehamn.	165

16	Julio	Reunión presencial Leganés	3	Definir requisitos finales de funcionamiento	-
31	Agosto	Reunión presencial Leganés	1	Firmar requisitos finales de funcionamiento	-
8	Septiembre	Reunión presencial Leganés	2	Explicación del sistema de MNC	-
23	Septiembre	Reunión presencial Leganés	2	Instalación de software de MNC	-

Tabla 15. Lista de hechos generales durante la duración del proyecto

- Los apartados en cuya columna aparece un guión (-) son aquellos cuyo coste viene indicado en otras secciones de este presupuesto.

TOTAL presupuesto para costes puntuales = 710,06 euros.

Presupuesto de costes fijos

Concepto	Coste/mes	Meses	Coste total
Alquiler apartamento individual	325,7	6	1954,2
Línea telefónica para internet	30	6	180
Alimentación apartamento	300	6	1800
Telefonía móvil (Sonera)	5	6	30

Tabla 16. Lista de costes fijos dados durante la duración del proyecto

TOTAL presupuesto para costes fijos = 3964,2 euros.

Presupuesto de ingresos para cubrir gastos en Finlandia (sufragios)

Día	Mes	Concepto	Cantidad (€)
7	junio	Ingreso de la beca Erasmus	1640

TOTAL presupuesto para sufragios= 1640 euros.

3.2. PRESUPUESTO DE COSTES ASOCIADOS A MATERIALES.

Concepto	Cantidad	Coste unitario	Coste Total
Vestuario para clima frío		300	300
Ordenador Portátil Dell Studio 15	1 para 6 meses	815	407,5
Microsoft Windows 7 home		Incluido	
Microsoft Visual Studio 2008		Licencia estudiante	
Microsoft Office 2007 student edition	1 para 6 meses	139	69,5
Microsoft Office Visio 2007		Licencia estudiante	
Papel	2000	0,01	20
Cartucho de impresora	1	24	24
Libros y documentación			74

Tabla 17. Lista de costes asociados a materiales para el proyecto

TOTAL presupuesto para materiales = 875,00 euros

3.3. PRESUPUESTO DE COSTES ASOCIADOS A RECURSOS HUMANOS.

Tarea	Rol	Días	Coste total €
1. Recopilación de información general para el proyecto	Ing. Sénior	5	812
2. Planificación y coordinación de tareas en el proyecto.	Ing. Sénior	2	324,8
3. Documentación e información de la plataforma .NET y C#.	Ing. Sénior	2	324,8
	Ing. Junior	10	980
4. Documentación acerca del diseño de interfaces con WPF.	Ing. Sénior	8	1299,2
	Ing. Junior	20	1960
	Diseñador	2	204,4
5. Diseño y desarrollo de la interfaz de la aplicación.	Diseñador	8	817,6
6. Desarrollo de la funcionalidad de la aplicación.	Ing. Junior	17	1666
7. Integración de interfaz y funcionalidad.	Ing. Sénior	4	649,6
	Ing. Junior	8	784
	Diseñador	3	306,6
8. Pruebas.	Ing. Sénior	2	324,8
	Ing. Junior	3	294
9. Documentación	Ing. Sénior	2	324,8
	Ing. Junior	8	784
	Diseñador	2	204,4

Tabla 18. Lista de costes asociados a recursos humanos durante el proyecto

*El coste de los trabajadores viene expresado con los impuestos incluidos.

Total de días trabajados:

- Ingeniero sénior: 25 (162,4 euros al día)
- Ingeniero júnior: 66 (98 euros al día)
- Diseñador: 15 (102,2 euros al día)

Total presupuesto en recursos humanos: 12.061,00 euros.

3.4. PRESUPUESTO FINAL DEL PROYECTO Y RESÚMENES

El presupuesto designado a las diferentes tareas, bienes y personas queda resumido de esta manera:

Presupuesto para costes puntuales = 710,06 euros.

Presupuesto para costes fijos = 3.964,2 euros.

Presupuesto para sufragios= 1.640 euros.

Presupuesto para materiales = 875,00 euros

Presupuesto para R.R.H.H. = 12.061,00 euros.

Presupuesto total = 19.250,26 euros.

Concepto	Coste (€)	%
Costes puntuales	710,06	3,7
Costes fijos	3964,2	20,6
Sufragios	1640	8,5
Material	875	4,5
R.R.H.H.	12061	62,7

Tabla 19. Suma de costes del presupuesto para el proyecto final

Ilustración 59. Gráfico circular de la suma de costes del presupuesto

PRESUPUESTO FINAL DEL PROYECTO:

Incluyendo impuestos y factores de riesgo y beneficio.

Concepto	Coste (€)
Presupuesto total	**19.250,26**
Riesgo (8%)	**1540,02**
Beneficio (14%)	**2695,03**
Subtotal	**23.485,31**
I.V.A. (16%)	**3.757,65**
TOTAL	**27.242,96**

Tabla 20. Presupuesto final del proyecto con impuestos

TIEMPO ESTIMADO TOTAL DEL PROYECTO: 5 meses, 17 días.

4. COSTE REAL DEL PROYECTO.

En esta parte de la memoria quiero explicar detalladamente lo que ha costado llevar a cabo este proyecto.
Al obtener una beca Erasmus para hacerlo en Finlandia, he contado todos los gastos e ingresos asociados a este propósito tales como el coste de la vida allí y otros.

4.1. COSTES ASOCIADOS A LA VIDA EN ÅLAND.

En esta sección explico todos los costes que han ido surgiendo durante la estancia de 6 meses en Finlandia mientras se desarrollaba el proyecto.

Costes puntuales:

Día	Mes	Concepto	Duración (h)	Concepto	Precio
5	Enero	Reunión skype	1	Internet mes	0,03
11	Enero	Viaje Madrid-Mariehamn	7,2	Viaje de ida a Mariehamn	165
24	Febrero	Propuesta de nuevo proyecto	0,5	Juan Llorens nos propone trabajar con MNC	-
26-28	Marzo	Viaje a Helsinki	72	Conocimiento general	27
8-10	Abril	Viaje a Tallin con clase de Technical english 3	72	Viaje obligatorio para la clase de T.E.3	80
15	Abril	Viaje a Madrid	216	Motivos personales	277
19	Abril	Reunión skype	1	Introducir tecnologías y definir requisitos	0,03
22	Abril	Viaje a Bruselas	48	Problemas con nube de ceniza	-

24	Abril	Llegada a Mariehamn	8	Problemas con nube de ceniza	-
27-30	Abril	Viaje a Turku (Finlandia)	96	Conocimiento general	24
4	Mayo	Presentación interfaz	2	Muestra de interfaz	-
7	Junio	Presentación interfaz 2	1	Segunda muestra de la interfaz.	-
15	Junio	Reunión Skype	1	Mostrando qué hemos hecho hasta entonces	-
22	Junio	Recogida de Pantalla táctil	1	Se me proporciona una pantalla táctil en sueco	-
27	Junio	Devolución de pantalla táctil	0,5	Devuelvo la pantalla táctil	-
28	Junio	Viaje Mariehamn-Madrid	8,5	Vuelta del periodo de estancia en Mariehamn.	125
16	Julio	Reunión presencial Leganés	3	Definir requisitos finales de funcionamiento	-
31	Agosto	Reunión presencial Leganés	1	Firmar requisitos finales de funcionamiento	-
8	Septiembre	Reunión presencial Leganés	2	Explicación del sistema de MNC	-
23	Septiembre	Reunión presencial Leganés	2	Instalación de software de MNC	-

Tabla 21. Lista de costes puntuales reales durante el proyecto

- Los apartados en cuyo coste podemos ver un guión (-) son aquellos cuyo valor contable aparece reflejado en otros apartados de este histórico de costes.

TOTAL costes generales = 717,90 euros.

Costes fijos:

Concepto	Precio mes	Meses	Precio total
Alquiler apartamento individual	325,7	5	1628,5
Alquiler apartamento compartido	162,85	1	162,85
Línea telefónica para internet	10	6	60
Alimentación	280	6	1680
Tickets restaurantes	50	6	300
Telefonía móvil (Sonera)	8	6	48

Tabla 22. Lista de costes fijos reales durante el proyecto

TOTAL gastos fijos = 3879,35 euros

Ingresos recibidos para sufragarme los gastos en Finlandia.

Día	Mes	Concepto	Duración (h)	Detalle	Cantidad (€)
14	diciembre	Ingreso de la beca del gobierno español		Beca del ministerio de educación.	3200
17	diciembre	Ingreso de la beca de la comunidad de Madrid		Beca de apoyo para becas Erasmus	1000
12	marzo	Obras de saneamiento	18	Trabajo parcial	180
7	junio	Ingreso de la beca Erasmus			1640

Tabla 23. Lista de ingresos reales para sufragar gastos

TOTAL sufragios= 6.020,00 euros

4.2. COSTES ASOCIADOS A MATERIALES.

En este apartado incluyo los gastos asociados al material que he tenido que comprar para vivir en Åland y poder desarrollar el proyecto.

Concepto	Cantidad	Coste unitario	Coste Total
Vestuario para clima frío		300	300
Ordenador Portátil Dell Studio 15	1 para 6 meses	815	407,5
Ratón para ordenador	1 para 6 meses	19	9,5
Microsoft Windows 7 home		Incluido	
Microsoft Visual Studio 2008		Licencia estudiante	
Microsoft Office 2007 student edition	1 para 6 meses	139	69,5
Microsoft Office Visio 2007		Licencia estudiante	
Microsoft Expression Studio 3	1 para 6 meses	599	299,5
Papel	2000	0,01	20
Cartucho de impresora	1	24	24
Libros y documentación			74

Tabla 24. Costes reales asociados a materiales

Total gastos de material = 1204 euros.

4.3. COSTES ASOCIADOS A RECURSOS HUMANOS.

En esta sección calculo lo que costaría mantener a las personas necesarias para este proyecto.

Tarea	Rol	Días	Coste(€)
1. Recopilación de información general para el proyecto	Ing. Sénior	8	784
2. Planificación y coordinación de tareas en el proyecto.	Ing. Sénior	4	392
3. Documentación e información de la plataforma .NET y C#.	Ing. Sénior	3	487
	Ing. Junior	15	1470
4. Documentación acerca del diseño de interfaces con WPF.	Ing. Sénior	6	974,4
	Ing. Junior	11	1078
	Diseñador	2	204,4
5. Diseño y desarrollo de la interfaz de la aplicación.	Diseñador	12	1226,4
6. Desarrollo de la funcionalidad de la aplicación.	Ing. Junior	19	1862
7. Integración de interfaz y funcionalidad.	Ing. Sénior	4	649,6
	Ing. Junior	6	588
	Diseñador	2	204,4
8. Pruebas.	Ing. Sénior	3	487,2
	Ing. Junior	5	490
9. Documentación	Ing. Sénior	3	487,2
	Ing. Junior	10	980
	Diseñador	2	204,4

Tabla 25. Costes reales asociados a recursos humanos

*El coste de los trabajadores viene expresado con los impuestos ya incluidos.

Total de días trabajados:

- Ingeniero sénior: 31 (162,4 euros al día)
- Ingeniero júnior: 66 (98 euros al día)
- Diseñador: 18 (102,2 euros al día)

Total gastado en recursos humanos: 12.569 euros.

4.4. RESUMEN Y COSTE TOTAL DEL PROYECTO.

Gastos generales: 717,90 euros.

Gastos fijos: 3.879,35 euros.

Sufragios: 6.020,00 euros.

Material: 1.204,00 euros

Gasto en recursos humanos: 12.569,00 euros.

Coste total del proyecto: 24.390,25 euros.

Concepto	Coste €	%
Costes puntuales	717,9	2,9
Gastos fijos	3879,35	15,9
Sufragios	6020	24,7
Material	1204	4,9
R.R.H.H.	12569	51,6

Tabla 26. Resumen de costes totales del proyecto

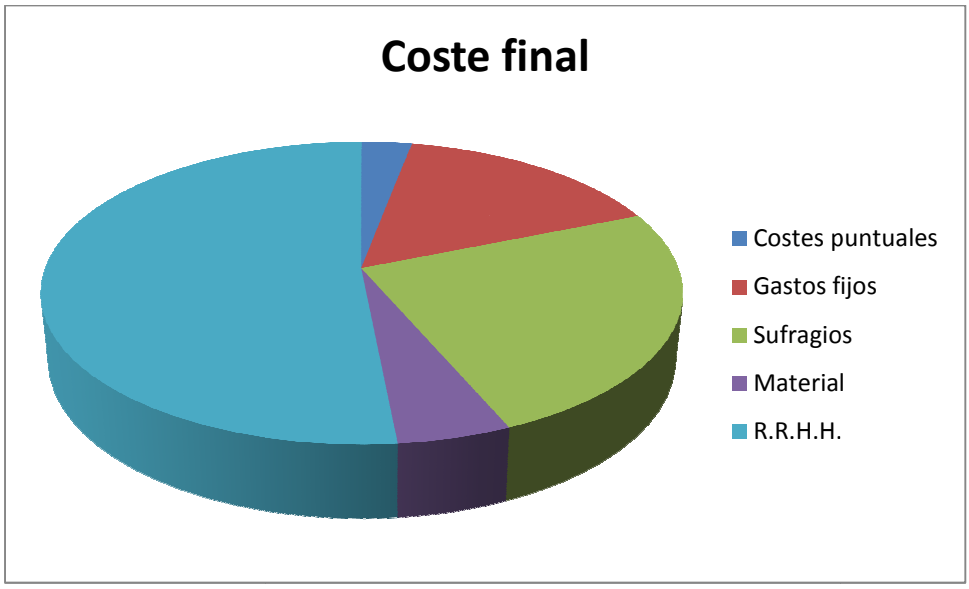


Ilustración 60. Gráfico asociado al resumen de costes totales

TIEMPO TOTAL DEL PROYECTO: 10 meses, 13 días.

COSTE FINAL DEL PROYECTO:

Incluyendo impuestos y factores de riesgo y beneficio.

Concepto	Coste (€)
Coste total	**24.390,25**
Riesgo (8%)	**1.951,22**
Beneficio (14%)	**3.414,63**
Subtotal	**29.756,10**
I.V.A. (16%)	**4760,98**
TOTAL	**34.517,08**

Tabla 27. Coste total final del proyecto con impuestos

TIEMPO TOTAL EN EL PROYECTO: 10 meses, 13 días.

5. CONCLUSIONES DE ESFUERZO.

Una vez expresados en los dos capítulos anteriores los desgloses de presupuesto y gastos, en este otro pretendo sacar una serie de conclusiones que demuestran los cambios entre lo estipulado y lo que finalmente ha ocurrido explicando su por qué.

5.1. ESFUERZO ECONÓMICO.

En este apartado voy a sacar conclusiones acerca de los costes que ha tenido el proyecto. Voy a trabajar sin incluir los factores de riesgo y beneficio o los impuestos como el I.V.A. puesto que son porcentajes que se añaden al final y no considero que sean procedentes en el estudio de los cambios entre el coste estimado y el coste final.

Costes puntuales de la vida en Åland

Presupuesto estimado = 710,06 euros.

Coste final = 717,90 euros.

Diferencia = 7,84 euros.

Como puede verse los costes puntuales finales son un 1,1% más elevado que el estimado, lo que concluye que estos costes fueron planeados de una manera eficiente.

La pequeña diferencia que radica entra las dos cifras se basa en cortas cifras como el recorte de gastos en viajes en contraposición con el incremento de coste asociado al problema que hubo al hacer el viaje Mariehamn-Madrid-Mariehamn debido a la nube de polvo que expulsó el volcán Eyjafjallajökull (Islandia) y que no me permitió volver a Mariahmn desde Madrid el día previsto, teniéndome que quedar más días y hacer una escala en Bruselas.

Costes fijos de la vida en Åland

Presupuesto estimado = 3.964,20 euros.

Coste final = 3879,35 euros.

Diferencia =- 84,85 euros.

El coste final es un 2,15% menor. Viendo el pequeño margen de error podríamos pensar que la estimación estaba bien hecha pero se han dado una serie de situaciones que han hecho que varíen ambas cifras hasta tener el resultado mostrado.

- Alquiler de apartamentos: Se estimó que viviría 6 meses sólo en un apartamento pero el último mes compartí los gastos con otra persona, reduciendo así 162,8 euros.
- Internet: Compartimos una línea gracias a un router wifi entre tres personas reduciendo los gastos 120 euros.
- Tickets restaurante: La universidad de Åland (Högskolan På Åland) nos daba 20 vales de 2,5 euros cada uno (50 euros al mes) para gastar en cualquiera de sus dos restaurantes asociados (Pizzería Fantasía o Café La Strada), lo que sumó al presupuesto 300 euros.

Por tanto, a pesar de ahorrar dinero en casi todos los costes fijos he incluido los tickets para comida que no previ que nos iban a dar, lo que hace que la diferencia entre la estimación y la realidad sea muy parecida.

Dinero para sufragar gastos

Presupuesto estimado = 1640 euros.

Sufragio final = 6.020,00 euros.

El dinero ingresado finalmente ha sido un 367,1% mayor que el estimado debido a:

- Becas: A la hora de irme a Åland busqué una serie de becas o ayudas económicas para poder vivir en Åland durante seis meses. En un principio sólo tenía asignada la beca Erasmus como viene en la estimación y que constaba de 1640, pero al haber aplicado a otras previamente y al irse resolviendo me fueron asignadas a su vez la beca general del ministerio de educación para estudios universitarios (3200 euros) y la beca de la comunidad de Madrid para estudiantes en el extranjero (1000 euros).
- Trabajo en Åland: un amigo nos comentó que necesitaba hacer una serie de reformas en su casa y para ello nos contrató. Finalmente el trabajo me aportó 180 euros.

Todo esto hace que la variación entre lo previsto y lo real sea un 367,1% mayor para lo real, lo cual a nivel personal me ha ayudado mucho.

Costes de materiales

Presupuesto estimado = 875,00 euros

Coste final = 1.204,00 euros

Finalmente el coste empleado en la compra de material ha sido un 27,3% mayor que lo estimado debido sobre todo a la necesidad de utilizar el programa Microsoft Studio Expression Blend 3 que se ha necesitado para diseñar la interfaz de la aplicación y cuya licencia ha tenido que pagarse.

Costes de Recursos humanos (R.R.H.H.)

Presupuesto para R.R.H.H.: 12.061,00 euros.

Coste recursos humanos: 12.569,00 euros.

Finalmente el coste empleado en el personal que ha participado en el proyecto ha sido un 4% mayor que lo estimado.

Esto, a pesar de que la diferencia en la duración del proyecto es muy grande (estimado 5 meses y 17 días y real 10 meses y 13 días), se ve compensado con el hecho de que las estimaciones dictaban 6 días más de trabajo para el Ingeniero Sénior y 3 para el diseñador, siendo ambos las personas que más salario cobran. Además hay que reseñar que no todos los días se trabajó puesto que hay grandes periodos de tiempo en los que se estuvo fuera de Finlandia y que fueron nulos para el trabajo como indico en los archivos adjuntos y en esquema de gastos generales.

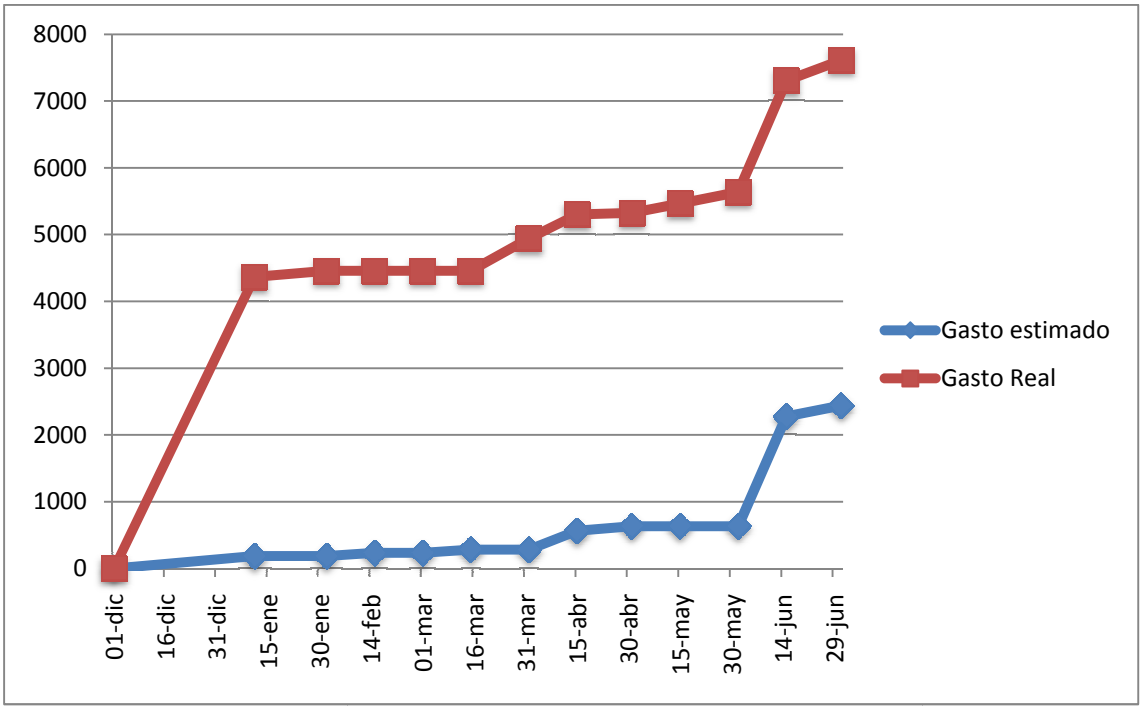


Ilustración 61. Gráfica resumen con diferencias entre coste presupuestado y real

Este gráfico que muestra la evolución de los costes estimado y real a lo largo de los 6 meses de estancia en Finlandia. Los valores que se han tenido en cuenta son los más relevantes tales como los costes puntuales (donde sobre todo influyen los viajes) y el dinero por sufragios. He decidido descartar los costes mensuales, los salarios y los costes de material puesto que los considero proporcionales a cada mes.

5.1.1. DIFERENCIA ENTRE EL PRESUPUESTO Y EL COSTE TOTAL DEL PROYECTO.

Presupuesto total asignado al proyecto:	19.250,26 euros.
Coste final del proyecto:	24.390,25 euros.

Como puede verse el coste final del proyecto sin sumarle los porcentajes de riesgo y beneficio o impuestos, ha sido un 14,30% mayor que el estimado (5140,25 euros) debido principalmente a que se han recibido 4.400 euros más para invertir en el proyecto por parte del gobierno Español mediante gastos para el sufragio citando la más importante, y al incremento en los gastos en materiales y en recursos humanos.

5.2. ESFUERZO TEMPORAL.

Tras desglosar los gastos e ingresos que hemos tenido a lo largo del proyecto, en este apartado quiero incidir sobre el tiempo que ha pasado en total e individualmente cada persona que ha participado.

Tiempo total que se estimó al proyecto: 5 meses, 17 días.

Coincide en su comienzo con el día que fui a Åland hasta el día que regresé y sumando otra serie de días para repasar y entregar.

Tiempo total que ha durado el proyecto: 10 meses, 13 días.

Igual que el tiempo estimado comienza el mismo día 11 de enero cuando fui a Finlandia. Sin embargo este ha sido más largo ya que se ha presentado una serie de factores que han retrasado el proyecto tanto y que explico a continuación:

- No se ha trabajado todos los días que eran laborales durante mi estancia en Finlandia. Al llegar a un sitio nuevo se requiere un periodo de adaptación y al ser la estancia relativamente corta (6 meses) hay que dedicarle tiempo a detalles que cambian continuamente.
- Muchas de las jornadas laborales no han durado lo estipulado, 8 horas. El apartamento y la biblioteca de la universidad han sido nuestros lugares de trabajo habituales. Estos ambientes de trabajo no suelen ser los más idóneos para realizar jornadas laborales largas.
- He realizado viajes que han restado mucho tiempo de dedicación al proyecto.
- La comunicación con el director de la empresa no ha sido posible siempre.

Estos motivos explicados han sido los que han alargado la fecha general de entrega del proyecto, sin embargo las estimaciones que hice de trabajo para los diferentes roles y personas considero que ha sido bastante acertada.

Ingeniero sénior: Estimé que tendría que trabajar 25 días y ha tenido que trabajar 31.

Ingeniero junior: Estimé que tendría que trabajar 66 días y ha tenido que trabajar 66.

Diseñador: Estimé que tendría que trabajar 15 días y ha tenido que trabajar 18.

Resulta impactante ver como la estimación de trabajo de las personas ha sido muy parecida a la real (tan sólo hay 9 días de diferencia sobre un total de aproximadamente 115) y el proyecto en general se ha alargado el doble (más de 5 meses). Esto es en mayor medida a motivos personales como los que he explicado anteriormente y que han contribuido profesionalmente de forma negativa.

SECCIÓN VI: EPÍLOGO

1. CONCLUSIONES.

Este capítulo puede dividirse en dos partes bien diferenciadas, la profesional y la personal.

Empezando con la primera puedo decir y valorar positivamente que si consideramos los principales objetivos propuestos, se han cumplido porque

1. Se monitoriza el estado de un ordenador y por tanto esto puede aplicarse a todos los de una red, no hay más que conectarlos hasta un máximo de 5 en el caso del paquete básico.

2. Se ha desarrollado una aplicación para pantalla táctil que accede a esos datos, modificando su apariencia según ellos. Con ello, un usuario podrá monitorizar los datos de los ordenadores conectados a la red.

3. Se ha trabajado con el framework .NET y se han desarrollado las aplicaciones en lenguaje C#, siendo siempre compatibles en código con los programas desarrollados con anterioridad por MNC.

4. El software creado ha sido integrado con el sistema con el que MNC trabaja. De este modo, la aplicación que obtiene la información de los ordenadores es ahora un plugin del programa con el que trabaja el propio software que MNC proporciona a sus clientes.

5. Se soluciona por tanto el problema que se le presentaba a MNC al no tener una aplicación de monitorización de ordenadores que ofrecer a sus clientes. Antes no podía comprobar el estado detallado de ellos y mucho menos que personal del propio barco lo controlara, ahora gracias a mi aplicación sí podrá.

Por tanto, a pesar de que se han desarrollado funcionalidades que en un principio se pedían y que suponían un proyecto mucho mayor tales como hacer los backups de la información o restaurar por completo un ordenador, se ha creado una base y una línea de trabajo para que sean implementadas en un futuro. De este modo no se tendrá que preocupar en diseñar un marco donde insertarlas, porque tanto Alejandro Alonso como yo lo hemos creado.

A nivel de conocimiento profesional quiero resaltar la formación que este proyecto me ha proporcionado y que considero cada día que pasa más importante en la vida laboral de un ingeniero de software:

1. Trabajo y familiarización con el entorno .NET. Sin duda son muchas las empresas que lo utilizan y yo he aprendido a manejarme teórica y prácticamente con él.

2. Aprendizaje de lenguaje C#.NET.

3. Aprendizaje de proyectos WPF. Esta ha sido la parte del proyecto que más me ha gustado, el desarrollo de la aplicación de pantalla táctil. El poder diseñar y contar con una tecnología tan novedosa y potente ha fomentado en mí el interés por el desarrollo de interfaces con mejores calidades. Además se me ha abierto un mundo como es el de diseño de aplicaciones software, que ahora está tan de moda sobre todo con la gran aceptación que tienen los tablet PC, los

smartphones y los sistemas operativos de MAC o Android que tanta importancia dan al apartado gráfico.

4. Uso de bases de datos. Realmente hemos implementado una base de datos que funciona para nuestro propósito. Se crea, se insertan y se consultan datos sin menor problema. A veces necesitas crear una aplicación real para darte cuenta de cómo funcionan las cosas reales; ese ha sido mi caso.

5. He aprendido a trabajar dentro de una empresa, a conocer los organigramas y los roles de las personas que en esta trabajan así como qué se puede esperar de cada uno. Ha sido absolutamente positivo.

Pero en este proyecto no todo ha sido fácil, al tener que incorporar este trabajo a una empresa real esperábamos más apoyo o soporte del que hemos recibido. La única persona que nos ha solucionado dudas, muchas veces relacionadas con una empresa de la que en un principio no teníamos nada que ver ha sido Miguel Ángel Zurdo, una persona con residencia en España con la que sólo nos hemos podido comunicar a través de internet mientras se desarrollaba el proyecto en España.

Tanto mi compañero Alejandro Alonso como yo esperábamos que la empresa asignara algo más de recursos a los proyectos que estábamos desarrollando para ellos, pero el resultado ha sido que MNC desde la dirección no ha respondido en muchas ocasiones a nuestras peticiones y la comunicación no ha sido fluida, lo que ha originado las siguientes consecuencias:

- Falta de prueba del software en una pantalla táctil. No se han podido desarrollar las funciones específicas de pantalla táctil tales como el zoom, por ejemplo.
- Falta de prueba del software en un sistema con 5 ordenadores conectados. Tan sólo se han podido probar las aplicaciones en determinados equipos con configuraciones diferentes así como en distintas redes de trabajo.
- Pobre especificación de requisitos en todas las áreas del proyecto.
- Aparición de numerosos problemas a la hora de integrar el software que se ha creado para este proyecto con el de la empresa. MNC está consolidada con un sistema determinado y yo, como desarrollador de software para este proyecto de fin de carrera desconocía por completo su funcionamiento. Desde la dirección de MNC no se han tomado las decisiones correctas para que estos proyectos fueran todo exitosos que podían haber sido.

Por otro lado soy consciente de que este software que hemos desarrollado, para ser implantado realmente en barcos a cargo de MNC, requiere un trabajo por parte de la propia empresa para instalarlo y acabar de acoplarlo a las instalaciones donde no hemos podido probarlo nosotros. Requerirá por tanto una cantidad de recursos, que dependiendo de los intereses de MNC, serán asignados o no.

A nivel personal quiero resaltar uno de los puntos más positivos de este proyecto como es el poder pensar que no se trata de un caso ficticio sino de uno real para una empresa real, y que por tanto ha sido creado cuidando los detalles al máximo. Realmente las aplicaciones creadas tienen que funcionar y ser estables para su buena implementación en los barcos que contratan los servicios de MNC. Esto ha sido posible gracias a Juan Llorens, quien como coordinador del proyecto nos ha brindado esta

maravillosa oportunidad, confiando en nosotros para llevarlo a cabo y trabajar junto a esta empresa, aunque luego la valoración final de esta sociedad corra por su cuenta. También creo que los problemas que hemos encontrado en el desarrollo de este proyecto ayudan a hacerse una idea de que el mundo laboral es muy diferente al universitario y gracias a ellos he podido aprender una importante lección.

Estos meses realizando este proyecto con MNC me han proporcionado no sólo una oportunidad única para hacer el proyecto de fin de carrera, sino que además me han aportado experiencia profesional en uno de los países con mayor prestigio profesional de Europa.

Más que una práctica de una asignatura cualquiera, este proyecto ha supuesto para mí la primera experiencia laboral en el campo de la informática.

2. FUTURAS LÍNEAS DE TRABAJO.

Este proyecto cumple los objetivos básicos, sin embargo ha habido algunas partes que se han quedado sin implementar, algunas detalles y otras grandes funcionalidades que habrá que insertar en el marco de he creado.

Referido a la aplicación de pantalla táctil **Touchable Screen Monitor Application** hay que trabajar sobre las siguientes líneas:

- Monitorización de otros componentes. Una vez se instale el programa o se tenga un entorno de pruebas adecuado, se podrá probar a acceder al router o modem con el que se acceda a la red, además de a servidores, RAID, etc. y que modificarán de un modo u otro la aplicación.
- Número de ordenadores monitorizados. Una vez se tenga un entorno de pruebas adecuado se podrá probar el trabajo de los 5 ordenadores a la vez, con el consecuente código a modificar y los fallos que se puedan dar.
- Diferentes paquetes de servicios con más ordenadores. Mi aplicación ha sido desarrollada para el paquete básico que consta de 5 ordenadores. Sin embargo, para poder monitorizar más ordenadores hay que rediseñar la estructura de las pantallas para que la presentación sea agradable. Es el único requisito costoso, por lo demás no hay más que copiar los procedimientos de un ordenador a los nuevos.
- Botones que llaman a funciones de la pantalla táctil. Por falta de entorno de pruebas no se han podido implementar funciones de acercamiento y alejamiento de pantalla. A pesar de esto se ven reflejadas en la interfaz para que se sepa donde insertar.
- Trabajo con la pantalla de login al sistema de MNC. En principio esta pantalla ha sido creada como punto destino de aquellas operaciones que no se pueden llevar a cabo en la aplicación pero que aparecen reflejadas de cara a un futuro.

En lo que se refiere al plugin **Detailed Info Plugin**, habrá que trabajar sobre todo en la monitorización de más parámetros de los que se monitorizan actualmente. Al no habernos especificado que parámetros querían que se monitorizara se han cogido como ejemplo algunos como el tamaño del disco duro, el espacio libre, la capacidad de los módulos de memoria RAM, familia o fabricante del procesador, estado de red… Siendo conscientes de que muchos de ellos son estáticos, es decir, que con el tiempo no varían su valor. Sin embargo, la explicación lógica a esta decisión se encuentra en que todos los dispositivos hardware que monitorizamos (disco duro, memoria física, procesador y red) son de vital importancia. Aunque no cambiarán sus parámetros, el programa habilita la posibilidad de detectar cuando uno de ellos no está funcionando bien y devuelve un valor que se sale del rango establecido, detectando un error entonces que podría ser muy grave ocasionando pérdidas de información o cuantiosos daños.

Por otra parte, se ha incluido un parámetro dinámico como es el espacio libre en el disco duro que se irá actualizando a medida que el software esté funcionando en el equipo, demostrando que a tiempo real el programa funciona.

Quizás MNC en un futuro quiera tener más información de la que proporcionamos ahora, de cualquier modo les hemos mostrado la manera en la que hacerlo, así que sólo tienen que utilizar otras clases WMI y seguir los procedimientos que en este proyecto se han desarrollado, teniendo un amplio abanico de posibilidades a su alcance. Pero como digo, eso es decisión de MNC.

Por otro lado, la creación de la base de datos ha sido muy productiva. A pesar de que hubo cierto escepticismo a la hora de usar SQLite tal y como recomendó Miguel Ángel Zurdo en vez de Access como usa MNC, se ha conseguido que la información esté ordenada dentro de un archivo en un servicio y que abre un amplio abanico de posibilidades totalmente compatibles con los procedimientos desarrollados por la empresa. Antes MNC no guardaba información de los ordenadores de los barcos donde instalaban el sistema. Ahora ya no sólo puede guardarla sino que puede tener un histórico de ella y así poder estudiar el comportamiento de los componentes que instale, de cara a una futura búsqueda de mayor rendimiento.

Otra futura línea de trabajo es la de la comunicación del servidor local del barco con la oficina de MNC en Finlandia, objetivo que atañe al proyecto de Alejandro Alonso y que está siendo llevado a cabo.

Pese a todo esto no creo que MNC deba limitarse a aplicar el software que hemos desarrollado en barcos. Monitorizar el estado de los componentes puede ser útil en otros campos empresariales. En cualquier empresa puede ser útil desde un ordenador donde esté un administrador de sistemas, monitorizar el estado de los ordenadores que tiene a su servicio.

3. ARCHIVOS E INSTALACIÓN DEL SOFTWARE.

3.1. ARCHIVOS NECESARIOS.

Para poder ejecutar todo el contenido de este proyecto se necesitarán lo siguiente:

- Sistema operativo Microsoft Windows Vista o Microsoft Windows 7.

- Framework Microsoft .NET 3.5.

- Instalador del programa proporcionado por MobileNetControl.

- Instalador del servicio llamado RepositoryManagementService para la interacción con el servidor.

- DetailedInfoPlugin con su archivo DLL asociado al compilarlo insertado dentro de la carpeta plugins del programa mencionado en el punto anterior.

- Touchable Screen Monitor application. Cuando se compile tendremos el ejecutable necesario.

3.2. INSTALACIÓN DEL SOFTWARE.

Para tener finalmente todo el proyecto instalado en el ordenador se tendrán que seguir los siguientes pasos, obviando que por defecto está instalado Windows 7 y el marco de trabajo .NET 3.5.

- Instalar el archivo Mobile Net Control ejecutando el instalador. Instalará el servicio llamado de la misma manera, necesario para la correcta ejecución de la aplicación.
- Instalar el servicio RepositoryManagementService para poder gestionar el intercambio de archivos con el servidor en un directorio diferente a aquel donde está instalada la aplicación MobileNetControl.
- Compilar el código del plugin “DetailedInfoPlugin” para que genere el archivo DetailedInfoPlugin.dll. Copiar dicho archivo a la carpeta de plugins del programa MobileNetControl.
- Ejecutar el código de la aplicación “Touchable Screen Monitor Application” para generar el ejecutable. Revisar la ruta de la base de datos donde se va a leer (archivos .cs de algunas ventanas) porque dependerá de la máquina donde esté instalado. Por defecto debería instalarse en C:\Archivos de programa\MobileNetControl Repository Manager, pero en los ordenadores de prueba la ruta es diferente.

3.3. EJECUCIÓN DEL SOFTWARE.

- Lanzar la aplicación MobileNetControl mediante el icono “Iniciar”. Existen algunos problemas con los permisos, de modo que se recomienda hacer esto con modo administrador.

- Iniciar el proceso RepositoryManagementService en caso de que no lo esté, aunque por defecto debería estarlo. Entonces el programa MobileNetControl empezará a detectar la información del ordenador, mandárselo al servidor y éste escribir la base de datos.

- Ejecutar el programa “Touchable Screen Monitor Application” y operar con ella, ya estará leyendo los datos de las computadoras (en este caso, en singular), del sistema.

4. CONTENIDO DEL PAQUETE.

Todo el proyecto, viene dividido y presentado en las siguientes carpetas:

0. Carpeta llamada "Memoria PFC Vicente García Adánez" y que contendrá:

- La memoria a color del proyecto en formato PDF.

1. Carpeta llamada "Touchable Screen Monitor Application", que a su vez contendrá las siguientes dos carpetas:

1.1. Carpeta llamada "Touchable Screen Application CODE" y que contendrá:

- Todo el código referente a dicha aplicación.

1.2. Carpeta llamada "Touchable Screen Application EXE" y que contendrá:

- Un acceso directo al archivo ejecutable .exe del código compilado que lanzará el programa directamente.

2. Carpeta llamada "Detailed Info Plugin", que a su vez contendrá las siguientes 2 carpetas:

2.1. Carpeta llamada "Detailed Info Plugin CODE" y que contendrá:

- Todo el códio del Detailed Info Plugin creado.

2.2. Carpeta llamada "Detailed Info Plugin DLL" y que contendrá:

- El archivo DLL generado al compilar dicho código y que supondrá el plugin a insertar en la carpeta del programa de MNC Mobile Net Control 2.0.

3. Carpeta llamada "Mobile Net Control 2.0 Installer" y que contendrá:

- El instalador de dicha aplicación.

4. Carpeta llamada "Mobile Net Control Repository Manager" y que contendrá:

- El instalador del servicio que permite acceder al servidor.

Esta página ha sido dejada en blanco intencionadamente.

Vicente García Adánez, autor del presente Proyecto Fin de Carrera titulado *Entorno de monitorización de sistemas informáticos embarcados mediante pantallas táctiles*, autoriza a que la información y documentación contenida en esta memoria pueda ser utilizada para la realización de otros Proyectos Fin de Carrera así como cualquier otra actividad docente sin ánimo de lucro.

Fdo. Vicente García Adánez